SIMULINK MODELING AND IMPLEMENTATION OF CMOS DENDRITES USING FPAA

A Thesis
Presented to
The Academic Faculty

By

Suma George

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in Electrical Engineering

School of Electrical and Computer Engineering
Georgia Institute of Technology
August 2011

SIMULINK MODELING AND IMPLEMENTATION OF CMOS DENDRITES USING FPAA

Approved by:

Dr. Paul E. Hasler, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. David V. Anderson
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Christopher J. Rozell
School of Electrical and Computer Engineering
Georgia Institute of Technology

ACKNOWLEDGMENTS

I would like to sincerely thank my advisor Dr. Paul Hasler for his constant guidance and support. Dr. Hasler's vision and work ethic are benchmarks that I shall always strive to achieve as I move forward in my career. I also thank my committee members, Dr. David Anderson and Dr. Christopher Rozell for their valuable inputs and comments. I would also like to thank all my lab-mates, especially Stephen Nease, Scott Koziol, Craig Schlottmann, Shubha Ramachandran and Stephen Brink who helped me during my research with their valuable suggestions and encouragement. Last but not the least I would like to thank my parents, my siblings Suja and Amrit and my best friend Sheahan for putting up with my busy schedule and supporting me during this extremely vital and enriching phase of my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

In this thesis, I have studied CMOS dendrites, implemented them on a reconfigurable analog platform and modeled them using MATLAB Simulink. The dendrite model was further used to build a computational model. I implemented a Hidden Markov Model (HMM) classifier to build a simple YES/NO wordspotter. I also discussed the inter-relation between neural systems, CMOS transistors and HMM networks. The physical principles behind the operation of silicon devices and biological structures are similar. Hence silicon devices can be used to emulate biological structures like dendrites. Dendrites are a branched, conductive medium which connect a neurons synapses to its soma. Dendrites were previously believed to be like wires in neural networks. However, recent research suggests that they have computational power. We can emulate dendrites using transistors in the Field Programmable Analog Array (FPAA). Our lab has built the Reconfigurable Analog Signal Processor (RASP) family of FPAAs which was used for the experiments. I analytically compared the mathematical model of dendrites to our model in silicon. The mathematical model based on the device physics of the silicon devices was then used to simulate dendrites in Simulink. An automated tool, *sim*2*spice* was then used to convert the Simulink model into a SPICE netlist, such that it can be implemented on a FPAA. This is an easier tool to use for DSP and Neuromorphic engineers who's primary areas of expertise isn't circuit design.

CHAPTER 1

INTRODUCTION

1.1 Neural systems and Analog VLSI

"The brain is a monstrous, beautiful mess. Its billions of nerve cells- called neurons- lie in a tangled web that displays cognitive powers far exceeding any of the silicon machines we have built to mimic it."

William F. Allman [1]

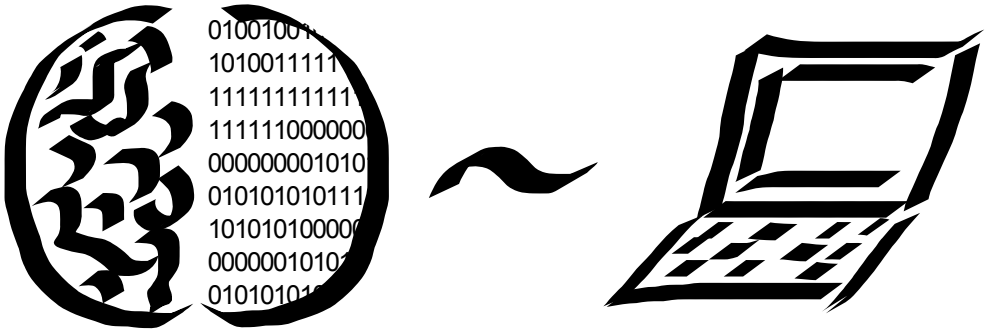


Figure 1. The human brain partly shown as an integrated circuit. The brain can be compared to a digital computer though it is much more computationally efficient.

The human brain, though studied a lot, is still a mystery when it comes to its functioning. Scientists are yet to determine what the intricate relationships and functions among various parts of the brain are. Merely knowing all the components of the brain doesn't lead to a sound understanding of how they interact with each other [2]. Numerous comparisons have been made between the brain and a digital computer with the biggest similarity being that they both process information. What sets the brain's neural networks apart is a very high computational efficiency, robustness and the ability to solve structured as well as ill-structured problems. The digital computer though has advantages of being very precise for well-structured problems. However, most real-world problems are not structured in nature.

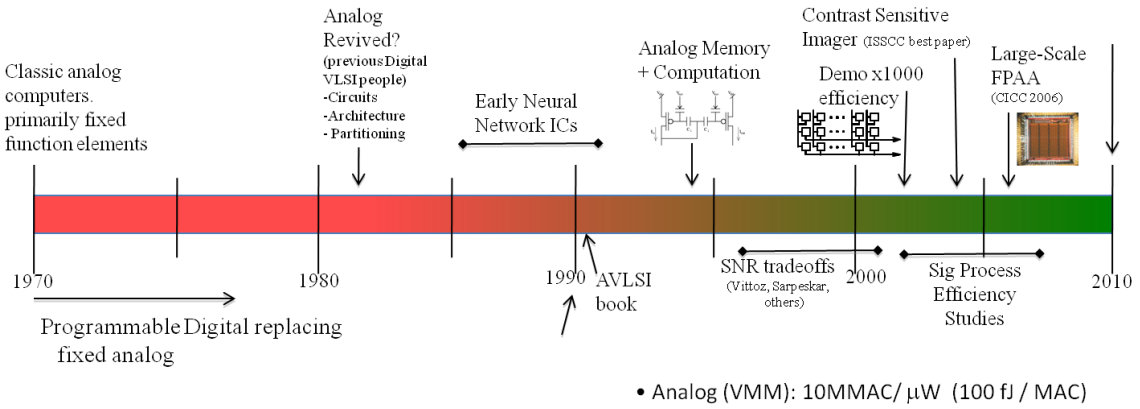


Figure 2. Analog Computation History. Here we see the use of analog in the computation path and how current day analog systems are programmable/reconfigurable. So what has led the resurgence of analog systems? It can be attributed to the considerably significant improvement in power efficiency, as well as improvements in terms of size or cost. The technology level required to work with typical digital system expectations has only appeared in the last few years. Current issues are familiarizing experts with application expertise to use this technology.

Even though a lot of progress has been made to develop systems that can tackle problems of natural language like the IBM Watson, we haven't yet designed a system that can function or adapt like the human brain.

One big flaw in this comparison between the brain and a digital computer is that neural computations are essentially analog in nature. It was observed that silicon devices have a similar physics to that of biological processes. Thus was born the field of neuromorphic engineering which was developed by Carver Mead. Mead suggested the use of VLSI systems to emulate biological systems because of their similar device principles. Hence, there is something that is intriguing about the nervous system that we should try and emulate if we want to develop systems that give us excellent performance with very low power consumption. Only recently we have acquired the technology to build reconfigurable and programmable analog systems.

In my thesis, I have focused primarily on a much smaller yet important component of the nervous system, the dendrites. Dendrites are essentially tree-like structures that connect neurons. A typical neuron in the nervous system has thousands of dendrites. Dendrites

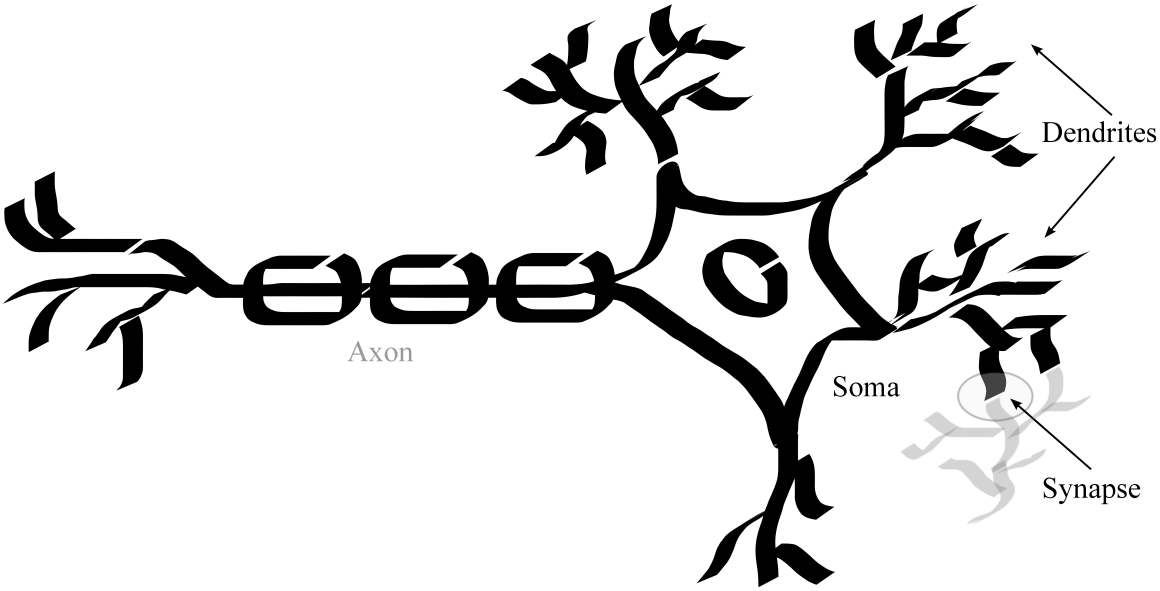


Figure 3. A pictorial representation of a Neuron.

receive a bulk of neural signals from other neurons via synapses. They serve as a post-synaptic site of a synapse and carry the signal to the soma.

Why is this an interesting problem to look at?

In both neural and silicon technology, active devices i.e. synapses and transistors occupy only up to $1-2\%$ of space. The remaining space is occupied by the 'wires' [2]. Studies have shown that in neural networks, dendrites don't merely act as wires but have interesting computational properties. This leads to very intriguing problems that we can address. First how do we model dendrites using CMOS transistors and can we use them to build computational models? So the obvious question arises as to why Analog VLSI systems are a close solution to building such networks. As discussed before, the basic underlying principles of operation for transistors in the sub-threshold region and biological systems are found to be similar in nature. Also essentially all real-world signals are analog in nature. However, only recently have we had access to technology that implements reconfigurable/programmable analog VLSI systems. These advances have made possible to build systems that are closer to their biological counterparts in terms of both efficiency and compactness.

Dendrites to the nervous system are equivalent to what a bus is to a digital computer.

Or so it seemed. Studies have showed evidence that dendrites are not merely just connecting wires but rather computational subunits that also contribute to the overall outcome of a neural operation [3]. Typically, dendrites have been modeled as passive linear cables. This classical model can be mathematically described by an equivalent RC delay line. The major predictions made by linear cable theory are based on this description. Analog VLSI circuits can be used to model this linear cable. If this hypothesis is correct, then the steady-state and dynamic behavior of both models should be qualitatively similar [4]. It has previously demonstrated the building blocks of a neural network, i.e. the channel, synapses and dendrites [5, 6, 4]. In this study, I demonstrate how a network of dendrites can be used to build a basic HMM classifier structure. We present simulation and experimental data for a single line dendrite and also results for a dendrite-based classifier structure. These structures can be used for speech and pattern recognition. The essential advantage of such a structure over digital implementation is low power consumption especially useful for implantable medical devices.

The chapters as we go forward are structured as follows. The second chapter focuses primarily on the tools that our lab has developed over the years which have enabled us to build reconfigurable analog systems and simulation models. It gives a brief overview of the tool flow. This will gives a better understanding of the experimental setup used. We have a comprehensive tool-set that includes Simulink blocks for building high-level systems as would be preferable for DSP and neuromorphic engineers and also the capability to write SPICE netlists for more experienced analog designers. The third chapter discusses the modeling and implementation of voltage-mode CMOS dendrites using reconfigurable systems. We have also developed a Simulink model for the dendrite that can be used to simulate its behavior as well as implement reconfigurable blocks on a chip. The simulation results of the block compared to experimental data are shown. The fourth chapter describes the use of dendrites as a computational block used for classification. I discuss the inter-relation

between neural systems, HMM networks and CMOS transistors. Experimental and simulation results for a single CMOS dendrite line are presented. I also demonstrate a YES/NO decision structure implemented using CMOS dendrites. The fifth chapter summarizes the work done and discusses the future possibilities and applications.

CHAPTER 2

FPAA AND TOOLS FOR FPAA DEVELOPMENT

In the previous section, I discussed how analog reconfigurable systems have been pivotal in developing biophysically inspired systems. In this chapter, we have a brief overview of all the tools used to make this possible. Our lab has over the years has developed a robust set of tools that makes this technology available to a larger group of engineers. These include the RASP family of FPAAs and also the software tool set consisting of *sim*2*spice*, GRASPER, RAT among others which has made using Field Programmable Analog Arrays (FPAA) easier.

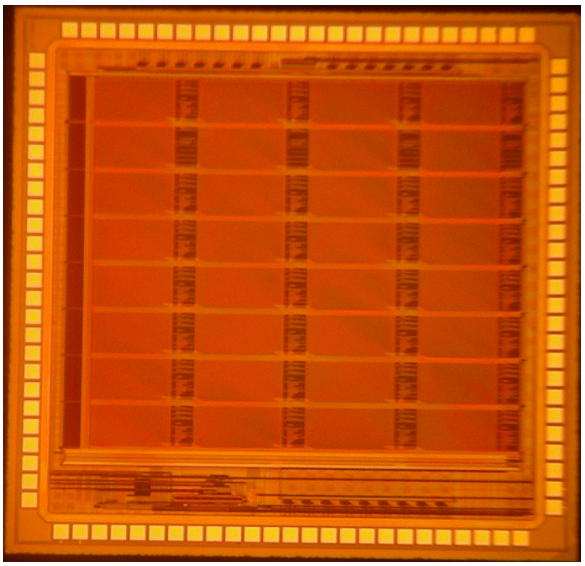


Figure 4. Tool set used: RASP 2.8a chip die photo which was used to get all the experimental data. Image reprinted from [7].

2.1 Field Programmable Analog Arrays: FPAAs

All of the data presented in this thesis comes from a reconfigurable hardware platform, the FPAA. The Field-Programmable Analog Array (FPAA) is a mixed-signal CMOS chip which allows analog components to be connected together in an arbitrary fashion. Reconfigurable Analog Signal Processor (RASP) was one of the first large scale FPAAs. It

allowed us to build multiple complex circuits. The specific chip used from the family of RASP chips for this research work is RASP 2.8a [7]. The RASP 2.8a is a powerful and reconfigurable analog computing platform that can be used to build Neuromorphic models. It consists of thirty-two Computational Analog Blocks (CABs). The CAB consists of groups of analog circuits which include nFETs, pFETs, Operational Transconductance Amplifiers, Capacitors and Gilbert multipliers among others. These act as the computational elements which together can form complex sub-circuits that can be used to build analog computational systems. The interconnection of the CAB components is achieved by the switch matrix. It essentially consists of floating gate (FG) pFETs. These $50,000$ programmable elements can be used not only as programmable interconnects for routing but also as adaptive computational elements. The switch matrix allows for both local routing between CAB elements as well as global routing. Last but not the least it has the programmer block, which selectively accesses a floating-gate device on the chip and through tunneling and injection tune it to on, off or operational in between. This is not only an efficient routing scheme but also can enable implementation of dense systems as we will see in further chapters. Examples of useful circuits implemented include vector matrix multipliers etc.

2.1.1 Floating gates

A floating-gate pFET's gate has no DC path to ground. The voltage to the gate is applied using a capacitive divider. This means that once charge is stored on the gate, it remains there without any need to apply potential directly. This also essentially means one less I/O pin. We can manipulate the charge on the gate using processes like tunneling and injection. To place charge on the gate, Fowler-Nordheim tunneling is used and charge is removed using hot electron injection.

The key principle behind the floating-gate technology was to build systems that can adapt and learn [8]. The most exciting aspect of implementing dendritic circuits using floating-gates is that it can be implemented in a very compact manner. As stated above,

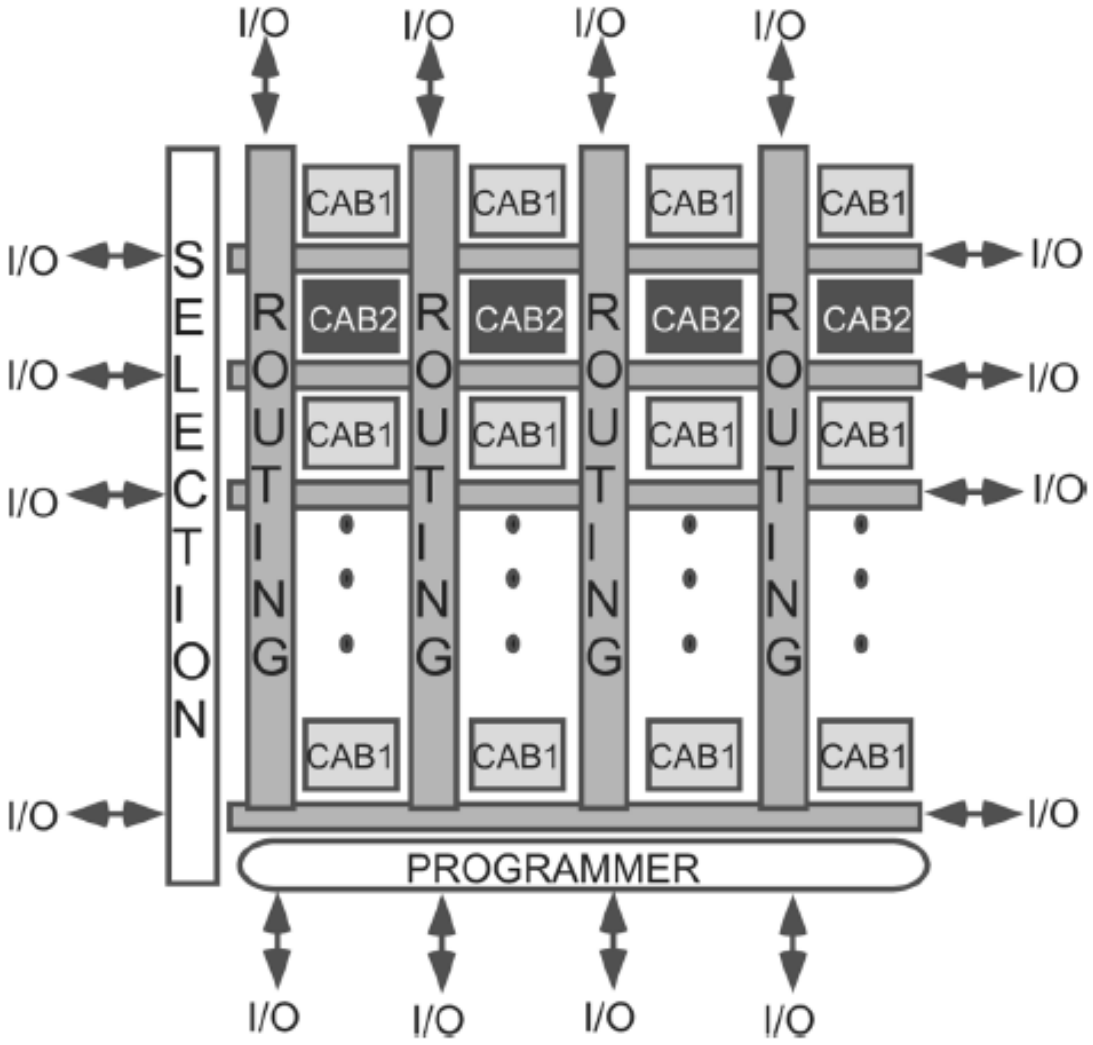


Figure 5. The RASP 2.8a CAB elements. Computational Analog blocks are interconnected using floating gate routing elements. Each floating-gate is programmed using a programmer and selection scheme. Analog input/outputs come in from all sides of the chip. Image reprinted from [7].

the switch matrix of the RASP 2.8a FPAA is completely made up of about 50,000 floating-gate elements. Thus huge arrays of dendrites can be made using the switch matrix. Its inherent function is to interconnect components which are similar to dendrites that are used to transmit signals from one structure to another.

Modeling dendritic circuits using floating gates is slightly more complicated than with regular FETs. The reason being the capacitive coupling from source and drain to the floating gate is more pronounced than regular pFETs [4]. Thus characterizing these coupling ratios is important if precision is desired. Another nonideality that arises due to indirect programming is the mismatch between the transistors that is 'programmed' versus the transistor that is actually used in the circuit. However recently, methods have been developed to characterize this mismatch [9].

As I have discussed before, floating-gates enable building very compact circuits. This enables building larger systems like HMM classifiers using CMOS dendrites. Considering the RASP 2.8a with thirty-two CABs, it is estimated that we can build 28 dendrites of length 24 and 4 dendrites of length 28. Also one must also take into account that neural systems are known to be inherently imprecise. Dendritic structures are not always similar and synapses are very unreliable. So one can say that this floating-gate mismatch is similar to dendrite-to-dendrite variability [4].

2.2 Simulink Tool: *sim2spice*

Engineers have conventionally relied on digital systems like DSPs and FPGAs to implement algorithms for signal processing. A lot of software tools are available that enable and simplify this process. Thus, existence of such intuitive software tools enables engineers to leverage the higher computational efficiency offered by hardware systems. Developing such high-level software tools for the FPAAs was thus a natural step. This would enable a wider community to design and implement analog systems easily. Our lab has developed *sim2spice*, which is a tool that automatically converts analog signal processing systems

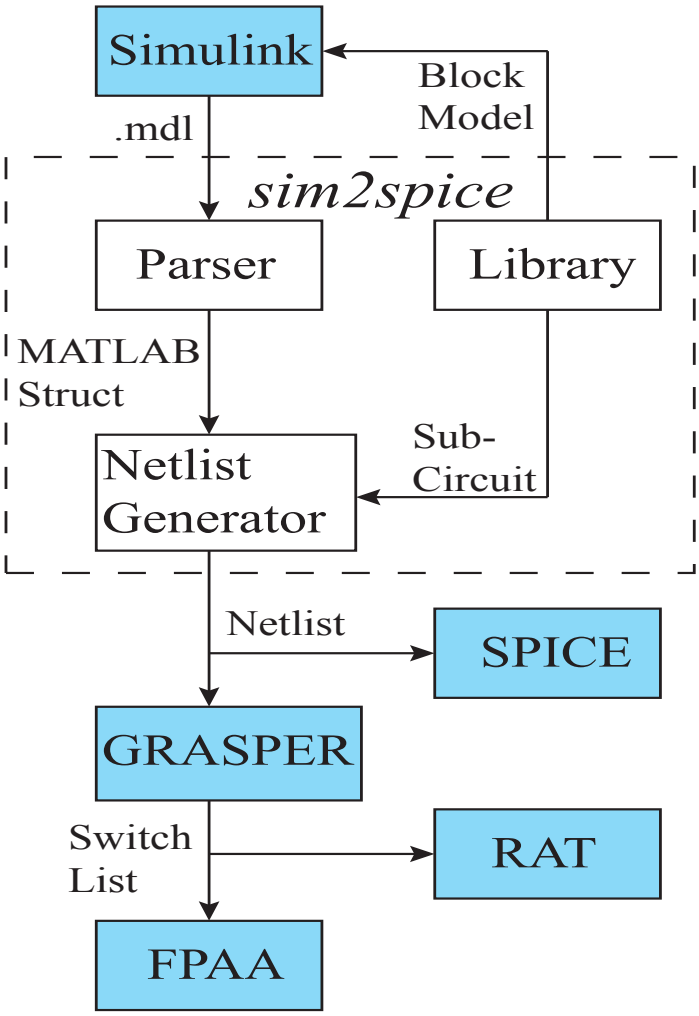


Figure 6. The tool flow used in experiments. Image reprinted from [10].

from Simulink designs to a SPICE netlist [10]. It is the top-level tool in a complete chain of automation tools that we will discuss further in this chapter. Multiple Simulink blocks have been implemented for different analog elements and circuits. The user has the choice of building his own analog system using basic analog elements or existing analog blocks to build a larger system. The basic analog elements consist of the CAB elements on the FPAA. All parameters of the block are configurable. Simulink also gives users the ability to encapsulate the final system created into a block; this ensures an ever-growing library of analog blocks that can be used. The simulink block mainly serves two purposes: First it converts the block-level simulink model into a SPICE netlist which can be implemented on

the FPAA. Secondly it can also be used to run a behavioral simulation of the circuit. These set of tools make it much easier for DSP and neuromorphic engineers who have little or no circuit design experience. A simulink block is defined by level-2 M file S-functions. It consists of mainly four files :

1. S-function simulink block : It consists of the physical dendrite block with its inputs,outputs and other input parameters that need to be defined. This is the user-interface block, where the user can configure the block. As illustrated in Fig 1a. The input parameters that the user can specify are given in Fig 1b.

2. Simulink(.m) behavior file: The mathematical model based on the device physics of the silicon devices is used to simulate dendrites in Simulink.

3. MATLAB(.m) build script : It builds the SPICE netlist for the block.

4. Description file(.desc) : Defines list of parameters needed by the parser.

The simulink tools provide an easier and flexible platform to develop complex systems. The behavioral modeling of systems further enables a user to simulate the functionality of a block. Going forward, it is a powerful tool as it furthers the outreach of these systems to a larger group of people to build analog systems.

2.3 SPICE to FPAA

Now considering we have the SPICE netlists generated using *sim*2*spice*, here is a brief overview as to how we compile this netlist onto the actual hardware. The next steps in the tool flow are to compile the SPCIE netlist onto the hardware. The GRASPER tool is used to place-and-route the netlist onto the FPAA. The Programming and Evaluation board with the programming interface tool are used to target the hardware. Also for purposes of debugging, we have the RAT tool that enables the user to view and edit the routing.

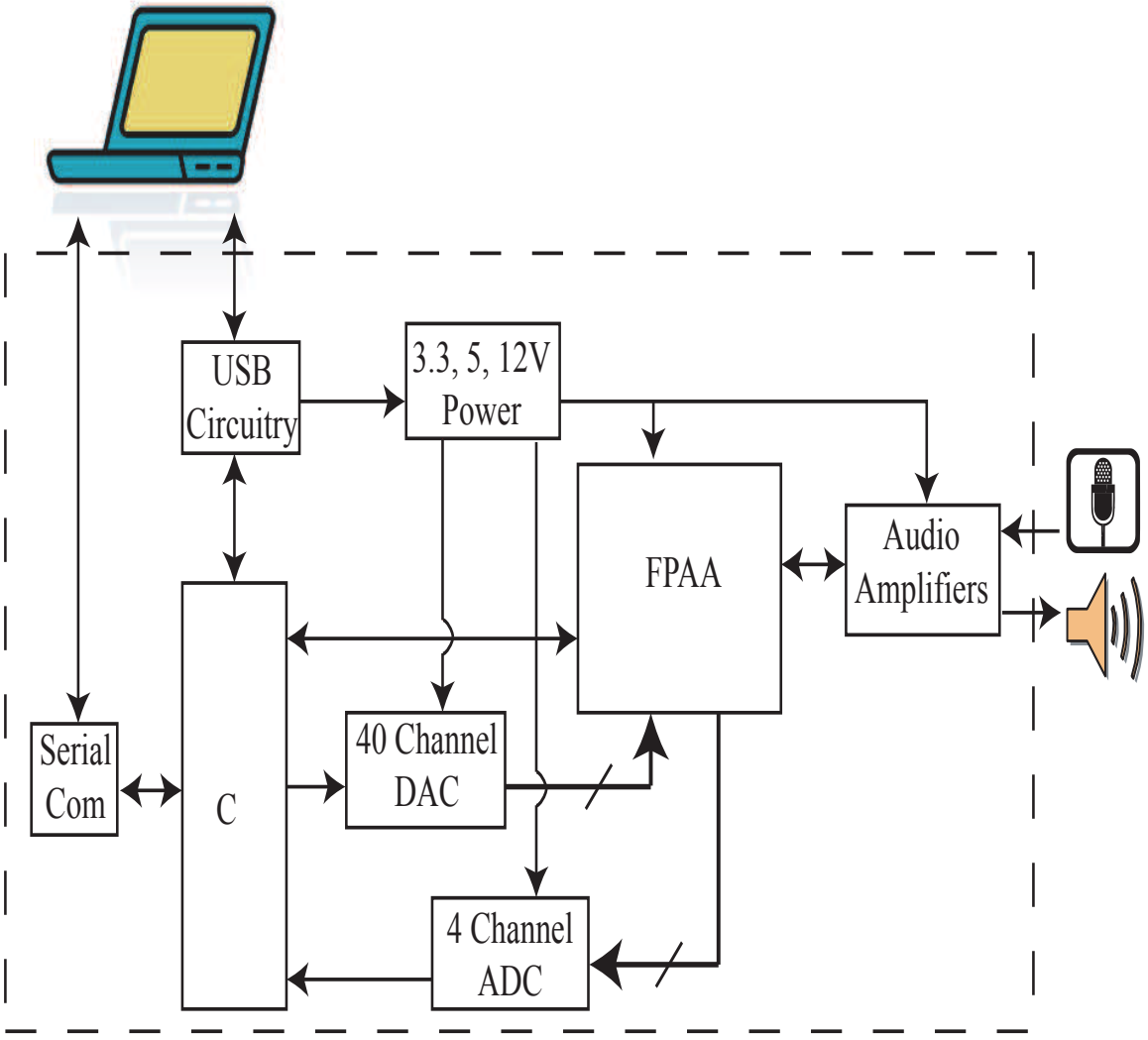


Figure 7. Overall system overview. Image reprinted from [11].

Place-and-route: GRASPER

GRASPER, developed by Baskaya et al., is the place-and-route tool used for targeting SPICE netlists to the FPAA [12]. The output is a list of switch addresses and the values to which they should be programmed, given in the format: (row, column, prog. value). The (row, col) address refers to the desired floating gates location in the crossbar matrix [13]. This tool can be used for the entire RASP family of FPAAs. One just needs to specify the particular FPAA used in the device (.dev) file. It contains all attributes specific to the given FPAA. Inclusion of this file ensures GRASPER is backward compatible with

previous generations.

RAT Visualization Tool

The RAT tool was developed by Koziol and Abramson. It provides a graphical way to view and edit the compiled circuits [11]. This tool comes in handy for designing and debugging on the FPAA. The input to the RAT is a programming (.prg) file that includes the switch list in the form output by GRASPER. The RAT tool GUI enables the user to view and modify the circuit. Once the design editing is finished the new design is output to a new .prg file.

Evaluation and Programming Board

The custom four-layer printed circuit board (PCB) used, was built to program, communicate with and test the RASP family of FPAAs. This evaluation board communicates over and is fully powered by USB, but it also has the capability to be powered by a 5-V dc supply and communicate over a serial connection. The board is controlled by an ATMEL ARM micro-controller for handling instructions from the computer using MATLAB commands. It also includes a 40-channel 14-b digital analog converter (DAC), a four-channel 8-b ADC, audio input/output amplifiers and jacks and all of the programming circuitry not already on-chip. In order to have maximum control and flexibility, almost every signal is pinned out to a header: all 52 FPAA I/O (4 to SMA connectors), the 40 DAC channels, four ADC channels and many of the microcontroller and programming lines. The plane is jumpered so that power measurements can be taken [13].

CHAPTER 3

MODELING AND IMPLEMENTATION OF VOLTAGE-MODE CMOS DENDRITES

3.1 Neuromorphic Engineering

Neuromorphic engineering was pioneered by Carver Mead in the late 1980s. The Neuromorphic engineer's thesis is that silicon emulates biology. Silicon devices and biological structures share similar physical principles of operation. This implies that silicon devices can be used to *emulate* neurobiological systems. The consequences of this statement are two-fold. First, we can use Neuromorphic circuits to emulate biological systems and second, we can use these systems to perform novel computation. We will explore in this chapter how we can simulate dendritic behavior and also develop systems that are capable of performing computation.

Computational neuroscientists typically rely on mathematical models to emulate biological processes using the digital computer. The advantage of using Neuromorphic circuits is that it enables simulation through physics which is common to both silicon devices and biological circuits. Also this is more suitable for larger simulation sizes. This allows for real-time emulation of the dense biological systems which is faster and more power efficient [4, 14].

The dendrite is one such neurobiological element which is an important computational structure. The dendrite is a highly-branched conductive medium that connects neuron's synapses to its soma as shown in Fig. 8. It was previously believed that dendrites had no computational value and were modeled as wires. However, recent studies have demonstrated otherwise. Our efforts entail not only simulating a dendrite but with the future goal of building computational models using dendrites [14, 3, 15]. Before we build computational models though, it is important that we verify the basic properties of dendrites using analog CMOS circuits.

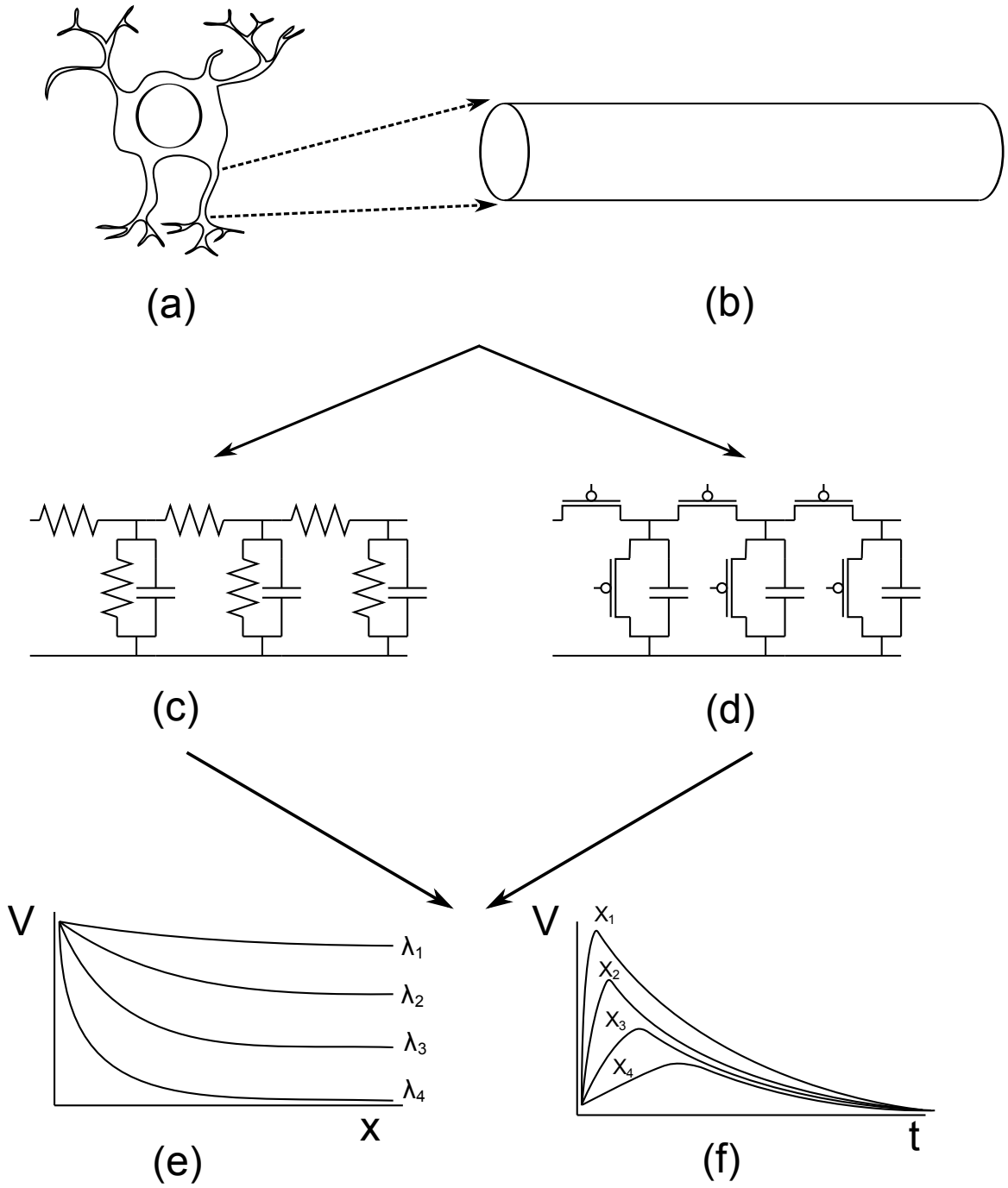


Figure 8. When operated in the correct regime, a VLSI dendrite model produces the behavior predicted by canonical linear models. (a) Dendrites are the structures which connect synapses to the cell body. They perform linear (and sometimes nonlinear) summations of input currents. (b) Neuroscientists typically model these structures as passive linear cables. (c) The classical model for this linear cable is an equivalent RC delay line. The major predictions of linear cable theory are based on this model. (d) An alternative model for the linear cable is a network of aVLSI elements, primarily MOSFETs and capacitors, where input currents are translated into small voltage signals which swing around a DC operating point. If (c) and (d) are equivalent, they should behave similarly. (e) The steady-state behavior of both models is expected to be an exponential decay in voltage, where the amount of decay depends on physical parameters. (f) The dynamic behavior of both models is expected to be exponential decay in space and a delay in time. Image reprinted from [4].

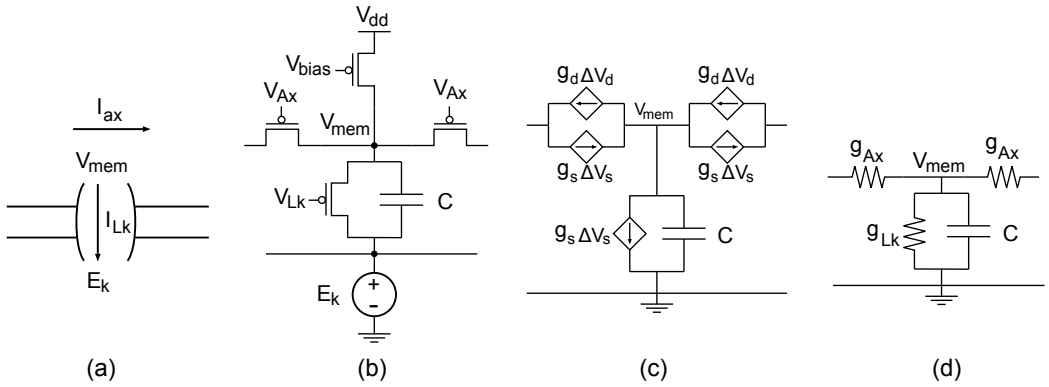


Figure 9. Various models of a dendrite. A biological dendrite is modeled as a conductive cylinder surrounded by an insulating layer. A cross section of this model is shown in (a), where I_{ax} represents the current flowing along the axial direction of the dendrite, I_{Lk} represents current from the dendrite to extracellular fluid through a leak channel and the external and internal potentials are V_{mem} and E_k, respectively. When we translate channels into transistors, we get the model shown in (b), where both the axial and leakage current flow through transistors. The external voltage is set by a voltage source E_k and V_{mem} is set by the bias structure. When we linearize the transistor model, the result is shown in (c) and (d). Current sources can be reduced simply to small-signal conductances. Image reprinted from [4].

3.2 Dendrites as Linear Cables

Dendrites have been modeled as linear cables historically. Their structure consists of a conductive solution, a phospho-lipid bilayer and ion channels. The conductive solution enables the current flow from the synapse to the cell body. The phospho-lipid bilayer separates the membrane potential from the external potential and the ion channels account for current leakage across the membrane. Wilfrid Rall adapted existing mathematics developed for core conductor cable and applied the same to dendrites. We have demonstrated that the behavior of a CMOS dendrite with pFET channels with small-signal inputs reduces to Rall's mathematical model. My basic thesis is shown in Fig. 9. In the model, we set the resting membrane potential V_{rest} by providing a bias current and assume that small signals as the inputs. This reduces our circuit to a linear model [4].

3.3 Introduction to Linear Cable Theory

The Linear Cable Model is one of the simplest models used to describe the function of dendrites. The dendrite can be represented as a conductive core enclosed by an insulating

layer. We can model the core as a long piece of resistive material. This can be discretized as incremental resistances R_{Ax}.

The simplest model neuroscientists use to describe the function of dendrites is known as the Linear Cable Model. The dendrite is treated as a conductive core surrounded by an insulating layer. The insulating layer is a phospho-lipid bilayer and it is modeled as a capacitance C because it separates the internal membrane potential from the extracellular potential. The leakage current can be modeled by a leakage resistance R_{Lk}.

Koch gives a simple derivation of the mathematical cable model for this circuit in [16]. If one writes down Kirchhoff's Current Law at the nodes V_{mem} and uses Ohm's Law $V = IR$ and the capacitor equation $I = C\frac{dV}{dt}$, then the following differential equation describes the system:

$$\lambda^2 \frac{\partial^2 V_{mem}}{\partial x^2} = \tau \frac{\partial V_{mem}}{\partial t} + V_{mem} - R_m I_{inj} \tag{1}$$

where I_{inj} is current injected into the dendrite, $\tau = R_{Lk}C$ and $\lambda = \sqrt{\frac{R_{Lk}}{R_{Ax}}}$. τ and λ are called the time constant and the space constant respectively. τ determines how voltages along the dendrite change with time and λ determines how voltages change with distance down the dendrite. If we only care about the steady-state solution, we can set the differential with respect to time equal to zero. This results in a solution for the steady-state behavior given in Eq 2 [4].

$$V(x) = V_0 e^{-|x|/\lambda} \tag{2}$$

3.4 Dendrite Simulink Block

For DSP and Neuromorphic engineers with little or no hardware experience, it is beneficial to have a software tool that can provide an easy interface with the hardware. MATLAB Simulink allows users to add new blocks with user-defined functionality and provides the user an interactive graphical interface. DSP engineers are familiar with this tool to a large

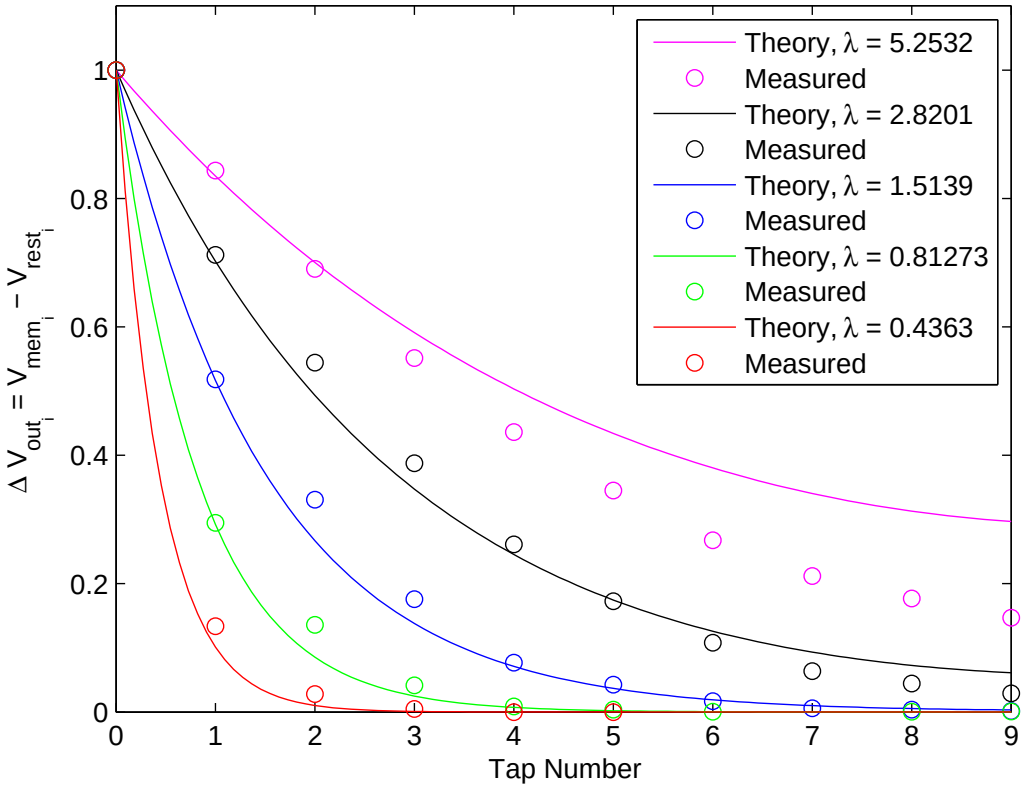


Figure 10. Steady-state decay of dendrite voltage. For five different values of λ, a ten-stage dendrite was biased at DC such that the V_{mem} nodes were about 20-50 mV above $E_k = 1$. Then a small DC current was injected into the first node. We then measured $\Delta V_i = V_{mem_i} - V_{rest_i}$ for every node in the dendrite. Then ΔV_i was normalized. The dots are experimental measurements and the lines represent how the voltages should decay if λ matches the theoretical value perfectly. The theoretical values essentially predict the "slope" of the logarithmic response and not the actual DC offset. This is why the normalized predictions are accurate for low values of stage number and seem to deviate as stage number increases. We're seeing error in the slope but not DC offset. The linear plot gives an intuitive physical feel of how the dendrite behaves.

extent. Keeping this in mind, we developed a Simulink model for dendrites. The Dendrite Simulink block provides users with a block-level interface. *sim2spice* [10] is the compiling tool we used to convert the block-level implementation to a SPICE netlist. The GRASPER tool [12] is then used to configure the FPAA and the RAT tool [11] is used to view and edit the routing.

The dendrite Simulink block is defined by level-2 M file S-functions and corresponding netlist elements. The elements used to model the block are the CAB elements on the FPAA. The input parameters for the block are configurable. The Simulink block can be used to

run a behavioral simulation of the CMOS dendrites and also generate a SPICE netlist to configure the FPAA. It consists of mainly four files :

1. S-function Simulink block: Consists of the physical dendrite block with its inputs, outputs and other input parameters that need to be defined. It is the user-interface block as illustrated in Fig. 11(a). The input parameters that the user can specify are given in Fig. 11(b).

2. MATLAB(.m) build script: Builds the SPICE netlist for the block

3. Description file(.desc): Defines list of parameters needed by the parser

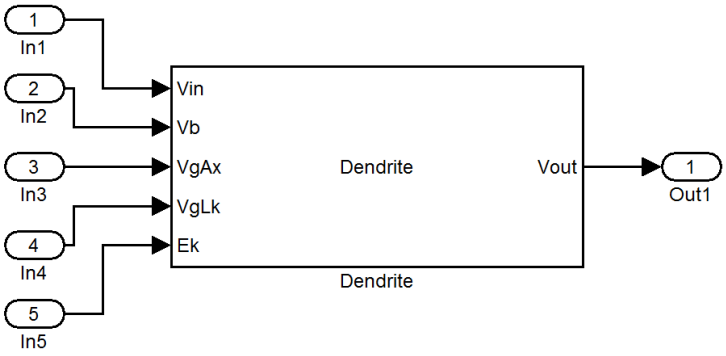
4. Simulink(.m) behavior file: Simulates dendrites in Simulink using the mathematical model based on the device physics of the silicon devices

3.4.1 Behavioral modeling

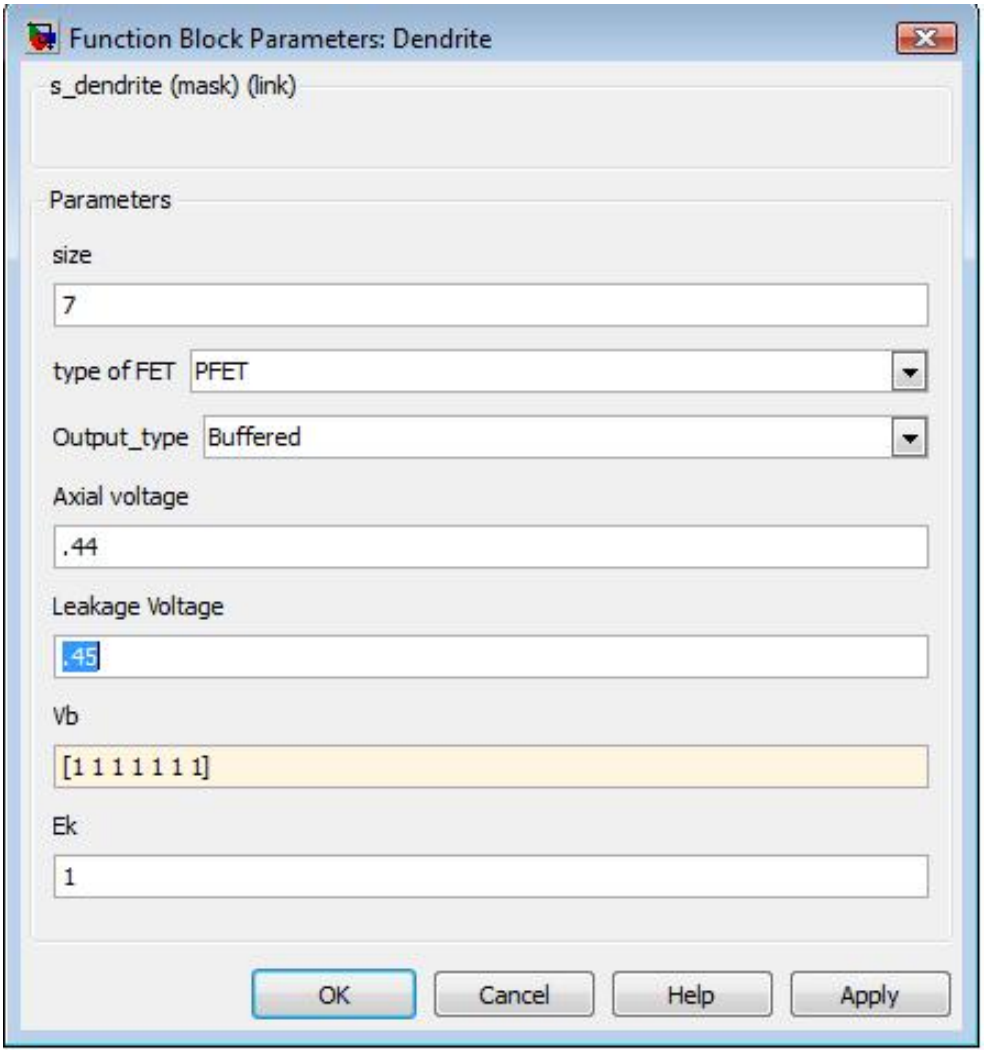
The Simulink block simulates the behavioral characteristics of the dendrite structure using the given inputs. This provides the user an insight to the working of the dendritic circuit when implemented using the FPAA. The MOSFET parameters used are based on the MOSFETS present on the FPAA. It is characterized by coupled ordinary differential equations (ODE) and solved using the ODE solver ode-45. The model has been tested for both static as well as time-varying inputs and has given reasonable results. I present below a detailed analysis of the mathematical model used, based on the device physics of silicon.

Consider a dendritic line as given in Figure 9, with n number of nodes. Current is injected only at the first node and the axial and leakage conductances are the same throughout. Applying KCL at node 1, the injected current and the bias current are the sum of the axial and the leakage currents. The leakage current comprises of the current through the leakage capacitor and the leakage transistor.

$$I_{inj} + I_{bias} = I_{Ax_1} + I_{Lk_1} + I_C \tag{3}$$



(a)



(b)

Figure 11. (a) Dendrite Simulink Block. This is a fully connected block with five inputs, which are the biasing voltages required for the dendritic line and the output port which denotes the voltage at every tap. (b) Block parameter window for the Dendrite Simulink Block. The window asks users to specify input parameters needed for the block. The user is asked to specify the number of stages, the type of FET used (PFET/FG-PFET), if the output should be buffered or not and the biasing voltages required for the circuit. Image reprinted from [4]

where,

$$I_{Ax_1} = I_0 e^{\kappa(V_{dd}-V_{Ax_1})/U_t}(e^{-(V_{dd}-V_1)/Ut} - e^{-(V_{dd}-V_2)/U_t}) \tag{4}$$

Now,

$$I_{Lk_1} + I_C = I_0 e^{\kappa(V_{dd}-V_{Lk_1})/U_t}(e^{-(V_{dd}-V_1)/U_t} - e^{-(V_{dd}-E_k)/U_T}) \quad + C\frac{dV_1}{dt} \tag{5}$$

Combining the above equations and substituting the values in equation 1,

$$\begin{aligned} C\frac{dV_1}{dt} = I_{inj} + I_{bias} - I_0 e^{\kappa(V_{dd}-V_A x)/U_t}(e^{-(V_{dd}-V_1)/U_t} - e^{-(V_{dd}-V_2)/U_t}) \\ - I_0 e^{\kappa(V_{dd}-V_L k)/U_t}(e^{-(V_{dd}-V_1)/U_t} - e^{-(V_{dd}-E_k)/U_t})) \end{aligned} \tag{6}$$

Applying KCL at node 2; the current through the first axial conductance and injected current equals the current through the second axial conductance, the leakage conductance and the leakage capacitance.

$$I_{inj} + I_{Ax_1} = I_{Ax_2} + I_{Lk_2} + I_C \tag{7}$$

We get,

$$\begin{aligned} C\frac{dV_2}{dt} = I_0 e^{\kappa(V_{dd}-V_A x_1)/U_T}(e^{-(V_{dd}-V_1)/U_T} - e^{-(V_{dd}-V_2)/U_T}) \\ - I_0 e^{\kappa(V_{dd}-V_A x_2)/U_T}(e^{-(V_{dd}-V_2)/U_T} - e^{-(V_{dd}-V_3)/U_T}) \\ - I_0 e^{\kappa(V_{dd}-V_L k_2)/U_T}(e^{-(V_{dd}-V_2)/U_T} - e^{-(V_{dd}-E_k)/U_T}) \end{aligned} \tag{8}$$

and so on and so forth for other nodes.

Generalizing the ODE for all nodes except the boundary cases, the voltage at the n^{th} node is given by

$$\begin{aligned} C\frac{dV_n}{dt} = I_{inj} + k_1(e^{V_{n-1}/U_T} - e^{V_n/U_T}) \\ - k_1(e^{V_n/U_T} - e^{V_{n+1}/U_T}) \\ - k_2(e^{V_n/U_T} - e^{E_k/U_T}) + I_{bias} \end{aligned} \tag{9}$$

where, C is the leakage capacitance and

$$k_1 = I_0 e^{((\kappa-1)V_{dd} - \kappa V_{Ax_n})/U_T} \tag{10}$$

$$k_2 = I_0 e^{((\kappa-1)V_{dd}-\kappa V_{Lk_n})/U_T} \tag{11}$$

This is a general expression for k_1 and k_2. However in our experiments V_{Ax} and V_{Lk} are the same throughout. The boundary conditions are as follows, At the first node,

$$\begin{aligned} C\frac{dV_n}{dt} = I_{inj} &- k_1(e^{V_1/U_T} - e^{V_2/U_T}) \\ &- k_2(e^{V_1/U_T} - e^{E_k/U_T} + I_{bias}) \end{aligned} \tag{12}$$

At the last node,

$$\begin{aligned} C\frac{dV_n}{dt} = I_{inj} &+ k_1(e^{V_{n-1}/U_T} - e^{V_n/U_T}) \\ &- k_2(e^{V_n/U_T} - e^{E_k/U_T} + I_{bias}) \end{aligned} \tag{13}$$

Writing the equations in vector form is useful as it reduces the time required for MATLAB computation. We define all the constants in the equations based on the MOSFETS used on the FPAA (κ, I_0,C) along with the input parameters as defined for the block (V_{Lk},V_{Ax},E_k). We can re-write these equations in vector form ,

$$\begin{aligned} \frac{d\vec{V}}{dt} = \frac{1}{C}(&a_1 \cdot I_{inj} + k_1(e^{a_2\cdot\vec{V}/U_T} - e^{a_3\cdot\vec{V}/U_T}) \\ &+ k_1(e^{a_4\cdot\vec{V}/U_T} - e^{a_5\cdot\vec{V}/U_T}) \\ &+ k_2(e^{a_6\cdot\vec{V}/U_T} - e^{E_k/U_T}) + I_{bias}) \end{aligned} \tag{14}$$

where,

$$\vec{V} = \begin{bmatrix} V_1 & V_2 & V_3 & \dots & V_n \end{bmatrix}$$

a_1,a_2,a_3, a_4, a_5 and a_6 are constant matrices whose size is dependent on the number of stages of the dendrite.

Results

I simulated a 10-stage dendrite using the Simulink Dendrite block. The nodes are biased at 1.02 V and a current is injected into the first node. The parameters used are $\kappa = .8626$, $I_0 = 10^{-18}A$, $E_k = 1V$, $V_{Ax} = 0.45V$ and V_{Lk} ranging between $.35V$ and $.5V$. These simulation

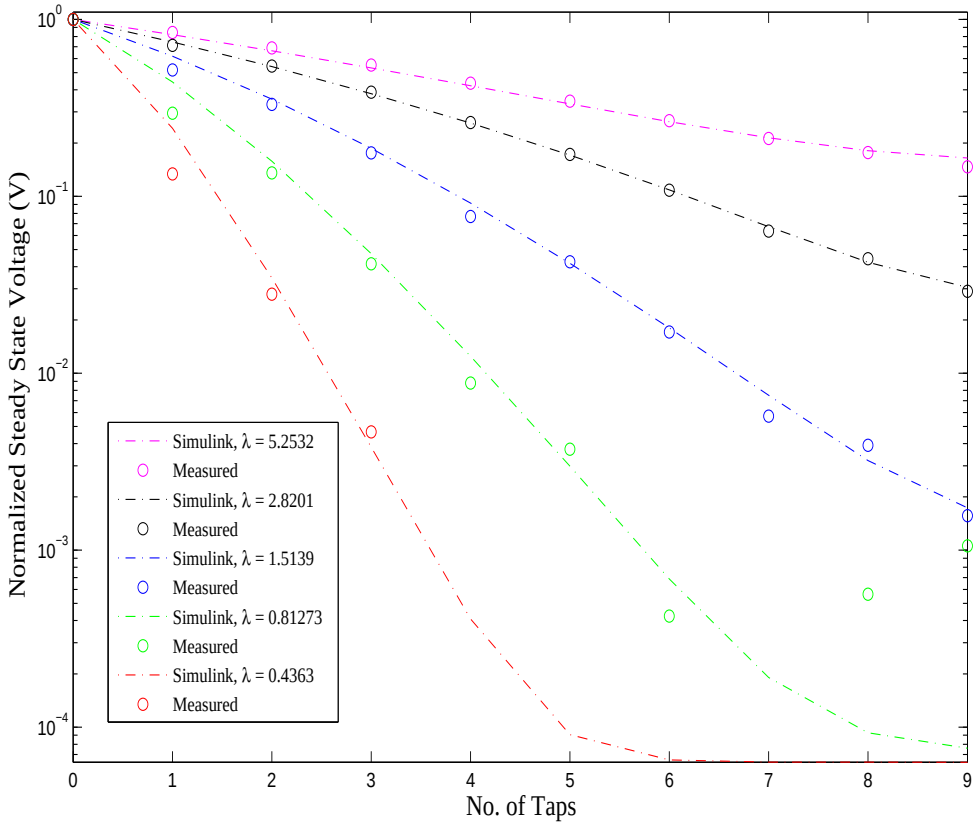


Figure 12. Comparing Simulation results to data obtained using the FPAA. After modifying the I_0 parameter in simulation to account for differences with the experimental setup (input current) and first-order non-idealities in the model, the two normalized curves have very similar qualitative behavior. Image reprinted from [4]

settings differ from our steady-state experiment in three ways. The input current is different from the experimental value; the node capacitance is higher than in the experiment and I_0 differs from the experimental I_0. The node capacitance should not affect the steady-state results and the value for I_0 helps offset first-order non-idealities in the model, as well as the differences in input current. Comparing the simulation with data obtained from the FPAA, I found that the two normalized results are qualitatively similar. The maximum percentage error was approximately 16.8. Refer Figure 12 for the results.

CHAPTER 4

LOW POWER HMM CLASSIFIER USING BIOPHYSICALLY BASED CMOS DENDRITES

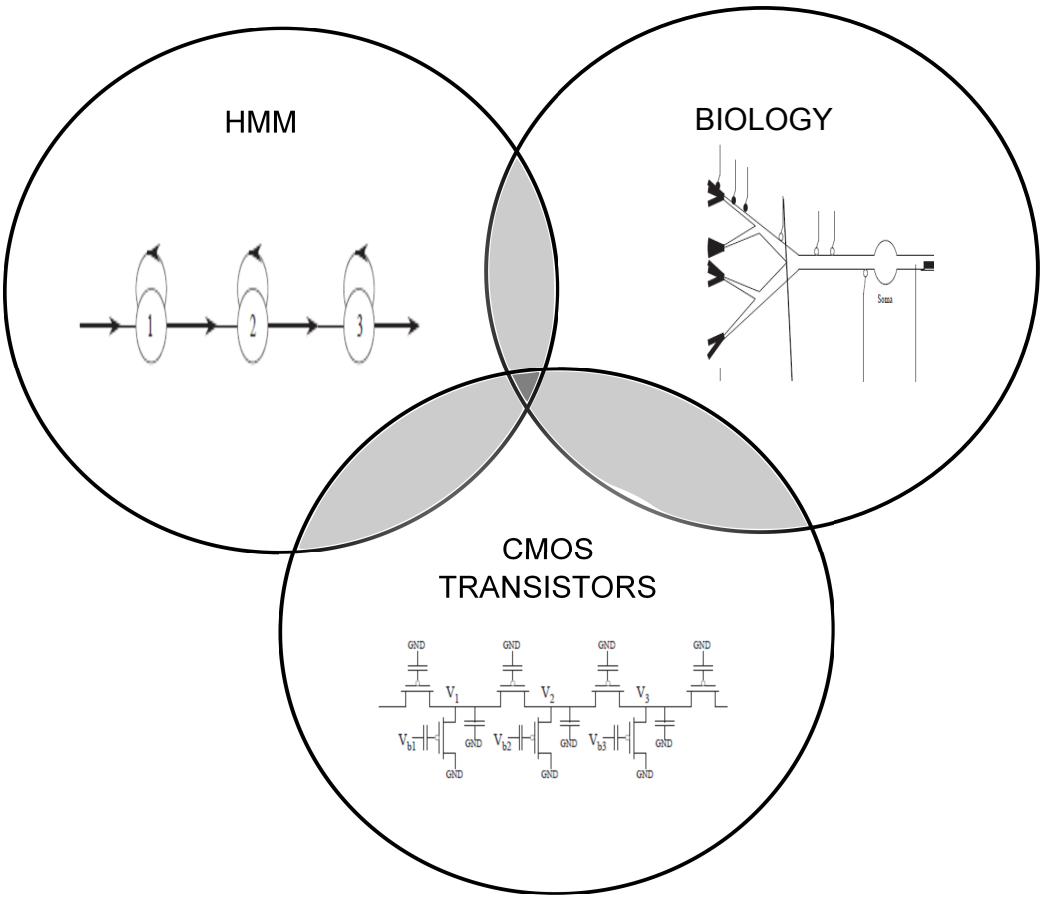


Figure 13. The Venn Diagram depicts the intersection between the fields of Neuro-biology, HMM structures and CMOS transistors. It has been demonstrated in the past how we can build reconfigurable dendrites using programmable analog techniques [4]. A dendritic network can also be used to build an HMM classifier which is typically used for speech recognition systems. Thus it is reasonable to believe that one can compare an HMM network with a group of cortical cells. The co-relations between these two areas is significant for many applications such as low-power implantable devices to aid hearing. Images reprinted from [17]

In the previous chapter I described the CMOS implementation of a dendrite. The next step was to build computational models using these circuits. Dendrites have been known to perform computations like coincidence detection [4]. It has been shown mathematically,

that dendrites are similar to a continuous-time HMM [17]. I have implemented a Hidden Markov Model (HMM) classifier for wordspotting using biophysically based CMOS dendrites. Wordspotting is the detection of small set of words in unconstrained speech [18]. It provides a simpler user-interface when used in voice control systems. In this chapter I show the results for a YES/NO HMM classifier. I shall also describe a simple HMM classifier model and its programmable IC implementation using CMOS dendrites. I shall also explore the co-relations between Neural systems, CMOS transistors and Hidden Markov Models (HMM). I will then discuss how a single dendritic branch is similar to an HMM branch and how it can be used to compute a metric. The HMM classifier is built using these dendritic branches, a Winner-Take-All (WTA) circuit and supporting circuitry. The system was implemented on a reconfigurable analog platform, the RASP 2.8a [7]. Experimental results for the same will also be shown. I will further discuss the advantages of such a system in terms of computational efficiency and the broader impact of such modeling.

HMM models are a popular choice for speech recognition systems. They have been known to be highly accurate. However, there is still no solution for wordspotting in unconstrained speech [19, 20]. Even though digital systems have greater accuracy than analog systems, analog systems have lower power consumption. This is closer to how biological systems function. Also, speech is analog in nature. Thus for certain applications, especially for implantable devices, an analog system is preferred [21]. Previously analog systems were not used much, as they were neither programmable nor reconfigurable. However, now that we have programmable/reconfigurable analog systems, building larger bio-inspired systems has become a reality.

Section 1 discusses the co-relation between neural systems, CMOS transistors and HMM networks. It also describes the similarities between a dendrite branch and an HMM branch. Section 2 further elaborates on dendrites as computational subunits. In Section 3 I will discuss Hidden Markov Models in more detail. In Section 4 I discuss the Single CMOS dendrite in detail. We will present experimental results for the single CMOS dendrite for

different parameters. I will also discuss the simulation model that I have developed and similar results seen. I will also present the mathematics which links HMMs to dendritic units. In section 5 we will discuss the tools that made the implementation of this classifier structure possible. In section 6 we will discuss the Analog HMM classifier implementation and discuss the experimental results seen. In section 7, I will discuss the computational efficiency of the system as compared to digital and biological systems. In section 8 I will discuss the broader impact of such systems. In the final section I will summarize the results and discuss the future possibilities.

4.1 Neuro-biology, CMOS transistors and HMM networks

It is an established fact that biological processes can be emulated using silicon devices. Neuromorphic engineers have modeled the channels, synapses, dendrites etc. using CMOS transistors [2, 6, 4]. These circuits aid our understanding of their biological counterparts. A bigger agenda in understanding biological processes is not just studying the elements themselves, but also understanding the intricate relationships they share [2]. This we hope will help us to understand how computation takes place in neural systems and build similar systems. Thus, an important part of this process is not only to build circuits that emulate individual biological elements but build computational systems using these circuits. Studies have shown that dendrites act as computational sub-units that contribute to overall computation of the neural network [3, 22]. It is then imperative that we build computational models using dendrites or say a network of dendrites. One such computational model is an HMM classifier used for speech recognition [17, 23]. Hidden Markov Models are predominantly used for speech recognition systems and many advanced speech recognition systems use HMMs and have up to 96% word accuracy [19].

Typically for speech recognition, short segments of the speech signals are analyzed and then the information is integrated for the entire word [21, 19]. The probability distribution b_i, represents the estimate if a symbol(short segment of speech/phoneme) was produced by

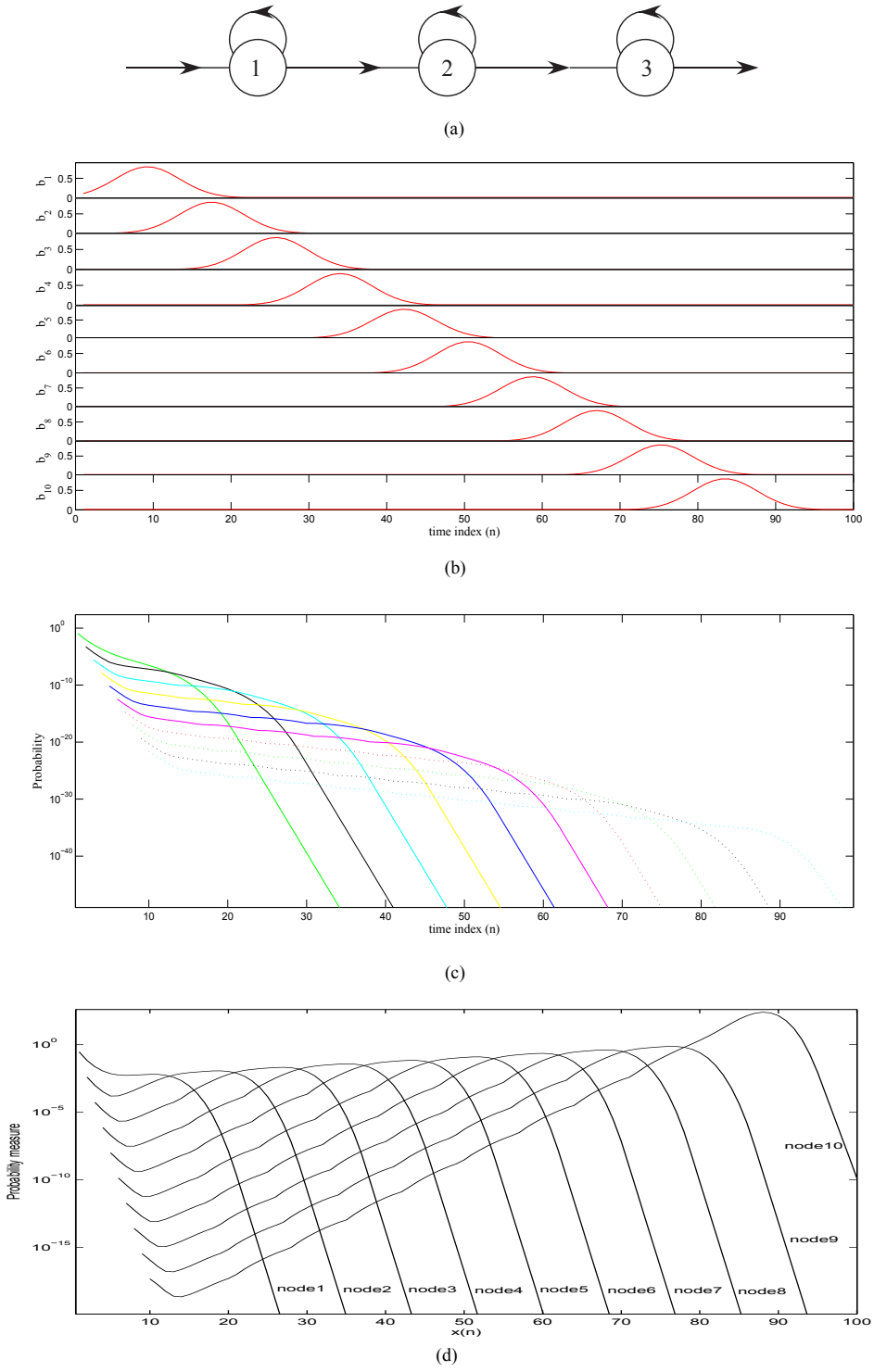


Figure 14. Simulation results for HMM state machine (a) Input probabilities varying with time (b) The probability distribution for each state of the HMM (c) Log likelihood outputs from all the states (d) Normalized log likelihood outputs from all the states. The outputs were normalized by multiplying them with an exponential function of the form $exp(t/\tau)$.

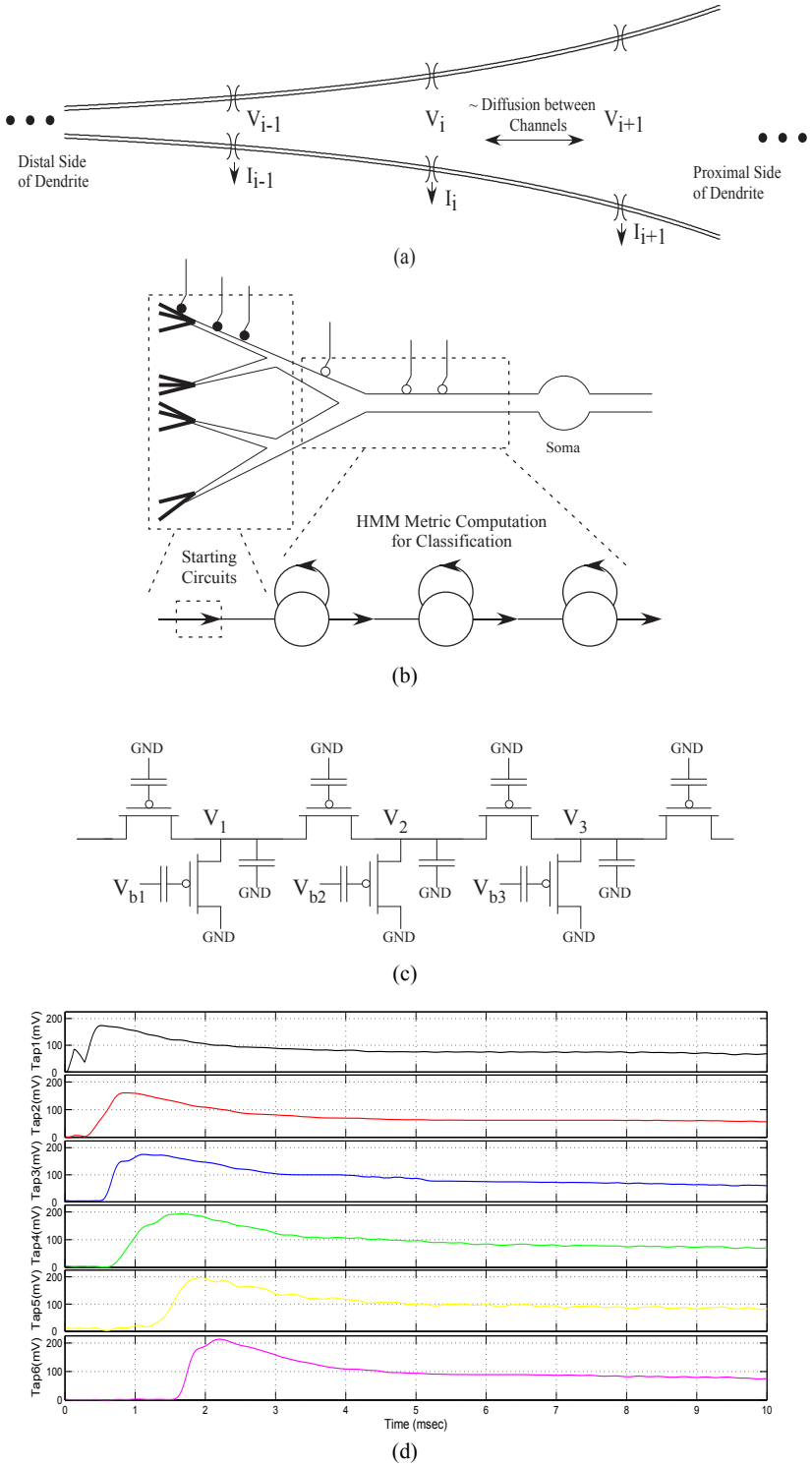


Figure 15. Co-relation of a dendrite branch to an HMM branch. (a) Dendrite with increasing diameter (b) Co-relation between basic HMM branch of state transitions and a dendrite branch. (c) Resulting IC implementation using programmable analog (Floating-Gate) circuits. Images reprinted from [5]. (d) Experimental results showing the outputs from each tap of the CMOS dendrite. These outputs are equivalent to normalized log likelihood outputs from the HMM states.

a state i. This acts as the input to the HMM state machine. Now, to group these symbols we calculate the likelihood of every state which is given by $\phi_i(t)$. This gives us an estimate of the likelihood that the particular state, was the end-state in a path of states that models the input signals [21]. For a typical HMM used for speech recognition the update rule is given by,

$$\phi_i[n] = b_i[n]((1 - a_i)\phi_i[n-1] + a_{i-1}\phi_{i-1}[n-1]) \tag{15}$$

Now a continuous-time Hidden Markov Model with a left-to-right topology has the following update rule,

$$\phi_i(t) = b_i(t)((1 - a_i)\phi_i(t-\tau) + a_{i-1}\phi_{i-1}(t-\tau)) \tag{16}$$

where, $b_i(t)$ is the input probability of symbol in state i and $\phi_i(t)$ is the likelihood of a state i for time t, τ is the time index between two consecutive time indexes and a_i is the transition probability between adjacent states. In a speech recognition model, the states are represented by the phonemes. It is interesting to note here, that even though the state sequence of such systems is implied; in continuous-time HMM one can't determine when the transition from states takes place. This is the reason why they are called '*Hidden*' Markov Models although the state sequence has a Markovian structure [19].

Studies have shown how a discrete-time HMM can be represented as a wave-propagating PDE that is continuous in time and space as given in (17) [23],

$$\underbrace{\tau\frac{\partial\varphi(x,t)}{\partial t}}_{\substack{\text{state}\\ \text{element}}} + \underbrace{\left(\frac{1}{b(x,t)} - 1\right)\varphi(x,t)}_{\substack{\text{decay}\\ \text{term}}} + \underbrace{a(x)\,\Delta\frac{\partial\varphi(x,t)}{\partial x}}_{\substack{\text{wave}\\ \text{propagation}}} = 0 \tag{17}$$

where, δ is the distance between two state nodes. This can be compared to analog diffuser circuits. Compare this to an RC delay line, based on which we model dendrites.

$$\underbrace{R_x C_i\frac{dV(x,t)}{dt}}_{\substack{\textit{state}\\ \textit{element}}} + \underbrace{R_x G_i V(x,t)}_{\substack{\textit{decay}\\ \textit{term}}} - \underbrace{(\Delta_x)^2\frac{d^2V(x,t)}{dx^2}}_{\substack{\textit{wave}\\ \textit{propagation}}} = 0 \tag{18}$$

$R(x)$ is the resistance, $G(x)$ is the conductance and C_i is the capacitance on node i. From (17) and (18) we can see the similarities in a continuous-time HMM and an RC delay

line [23]. The CMOS dendrites are modeled as an RC delay line only using CMOS transistors instead of the resistances. The above equations establish a co-relation between a continuous-time HMM model and a CMOS dendrite. As we have established the similarities between biology and silicon devices and between continuous-time HMM and CMOS dendrites; we can postulate that there is some inter-relation between HMMs and neural systems.

4.1.1 Dendritic computation and the HMM branch

As discussed before, many advanced speech recognition systems use Hidden Markov Models. These systems consist of a probabilistic state machine and a way to trace the most likely path through the state machine that produces the input speech signal. For this we estimate the input probability for a state, $b_i[n]$. It is the probability that signal in frame n is produced by state i. Now to integrate this information over an entire word, we determine the likelihoods $\phi_i[n]$ for the state machine. The likelihood of a state determines if it is the end-point of the most-likely path for a particular signal[21]. In Fig. 14 and Fig. 15 we see the similarities between the implementation of an HMM model for a wordspotter and a dendritic branch. For an HMM state machine for speech recognition, the inputs $b_i[n]$ are essentially Gaussian in nature [21]as seen in Fig. 14(a). Based on this we calculate the likelihood $\phi_i[n]$ for each state of the HMM branch. Now as can be seen in figure Fig. 14(b), the likelihood outputs for each state show a very a sharp decay and have a very high dynamic range. Thus to limit this range we normalize this output with an exponential function. It is interesting to observe that the normalized likelihood are essentially similar to EPSP signals. Also similar to an HMM branch, the output of the last node of the dendrite determines if we get a winning metric or not. These similarities lead us to hypothesize that an HMM state machine is similar to a dendritic branch. Let's explore this further.

An HMM branch and a dendrite branch have similar looking topologies. The HMM state machine used for wordspotting, as shown in Fig. 14(a), is a left-to-right model. This implies state transition is only possible left-to-right. The biological dendrite doesn't have a

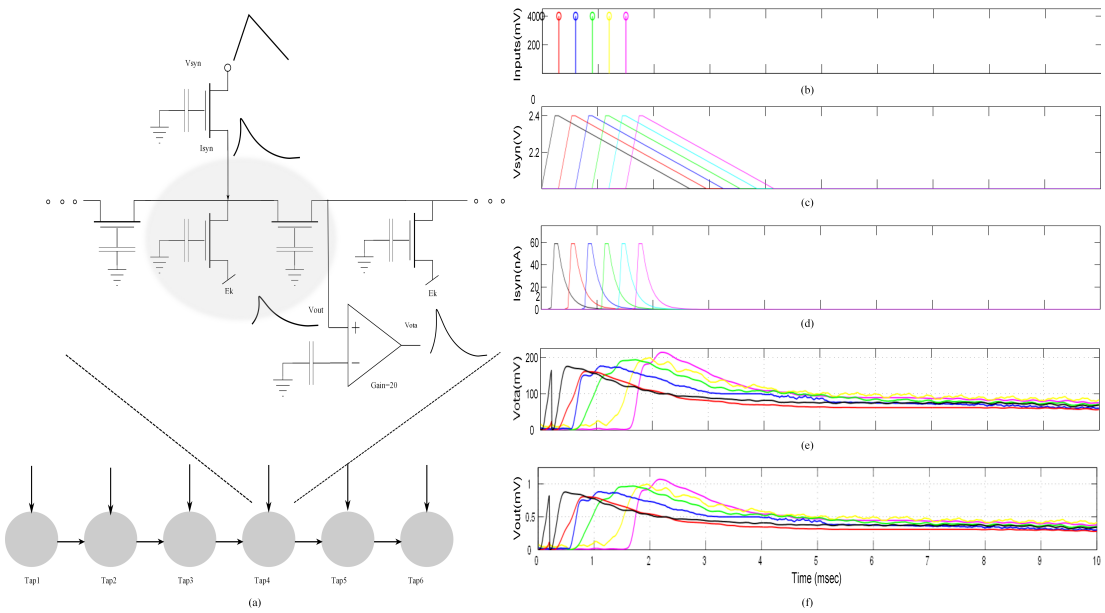


Figure 16. A step by step overview of a dendritic branch. On the left is a more detailed structure for a dendritic line and on the right are all the relevant signals as marked on the diagram (a) Detailed diagram for a single dendritic line which is equivalent to an HMM branch(b) The representation of input voltage on the source of the transistor representing the input synapses (c) The triangular input voltages V_{syn} at the source of the transistor representing the input synapses (d) I_{syn}, the input synapse currents into each of the different nodes (simulated). (e) V_{ota}, the output as seen from the FG-OTA. The gain of the FG-OTA is approximately 20. (f) V_{out}, the output voltage at each node.

constant diameter. Its diameter at the distal end is smaller as compared to the proximal end as shown in Fig. 15(a) and Fig. 15(b) [5]. Thus, for a similar CMOS dendritic line that is uni-directional, we would expect the axial conductances of the line to increase from left-to-right as shown in Fig. 15(c). This is the case of a dendrite with '*tilt*'. A '*tilt*' signifies a changing diameter through the length of a dendrite. Such a topology ensures that the current flow is in one particular direction i.e left-to-right. This also favors coincidence detection in the dendrite. Now for a single n-stage dendrite with tilt, with EPSP inputs in a sequence at subsequent nodes, the output observed at the end of the line was similar to the normalized likelihood outputs of an HMM classifier as shown in Fig. 15(d) and Fig. 14(d). This similarity is the premise of our reasoning that a left-right HMM state machine branch is similar to a dendritic branch. It means one can build a computational model using dendrites for classification. We will further elaborate on the mathematics behind this premise in the following sections.

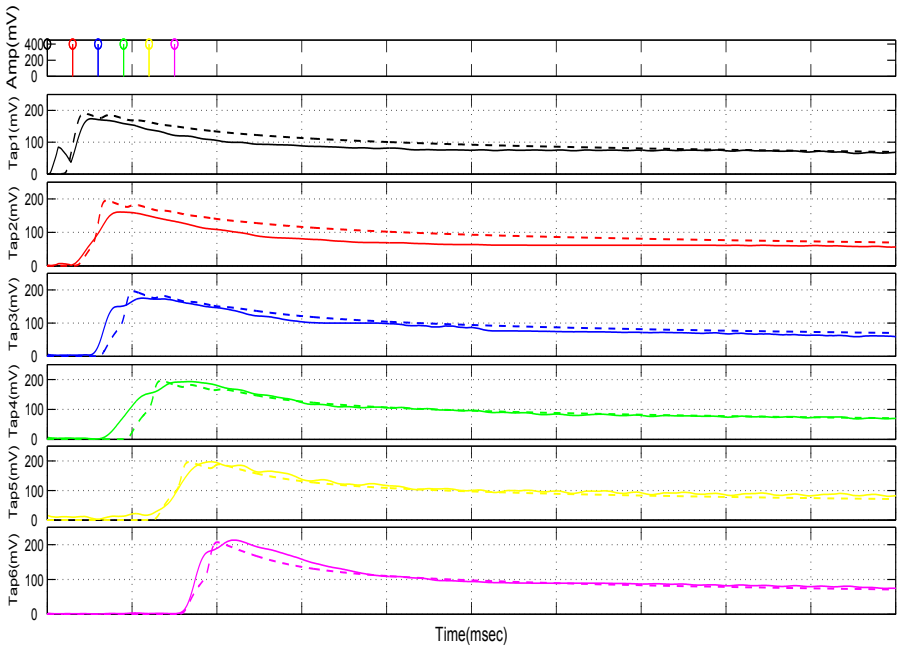


Figure 17. Simulation Data vs. experimental data comparison for a single CMOS dendrite. The dotted lines depict the simulation data and the solid lines are the experimental data. The parameters for simulation data are $V_{Leak} = 0.5V$, $V_{axial} = 0.5V$, $\kappa = 0.84$, $I_0 = 0.1fA$, $C = 1.3pF$, $E_k = 1V$ and $V_{dd} = 2.4V$.

4.2 Dendrites: Computational Subunits

Dendrites are highly branched tree like structures that connect neuron's synapses to the soma. They were previously believed to act just like wires and have little or no computational value. However, recent studies have shown otherwise [3, 17, 4, 16, 14]. Dendrites are believed to be computational subunits which perform some inherent processing that contributes to overall neural computation [3, 22]. Neuromorphic engineers have claimed that biological processes can be emulated using transistors. It has been demonstrated how silicon can be used to emulate biological processes as they share the similar physical principles. We have previously verified the basic properties of single line dendrites using analog CMOS circuit models [4]. These CMOS dendrites have been modeled to behave like biological dendrites and we have implemented them using both reconfigurable analog systems and a Simulink model. It is imperative then, that we build circuits that share the computational properties of neurobiological circuits.

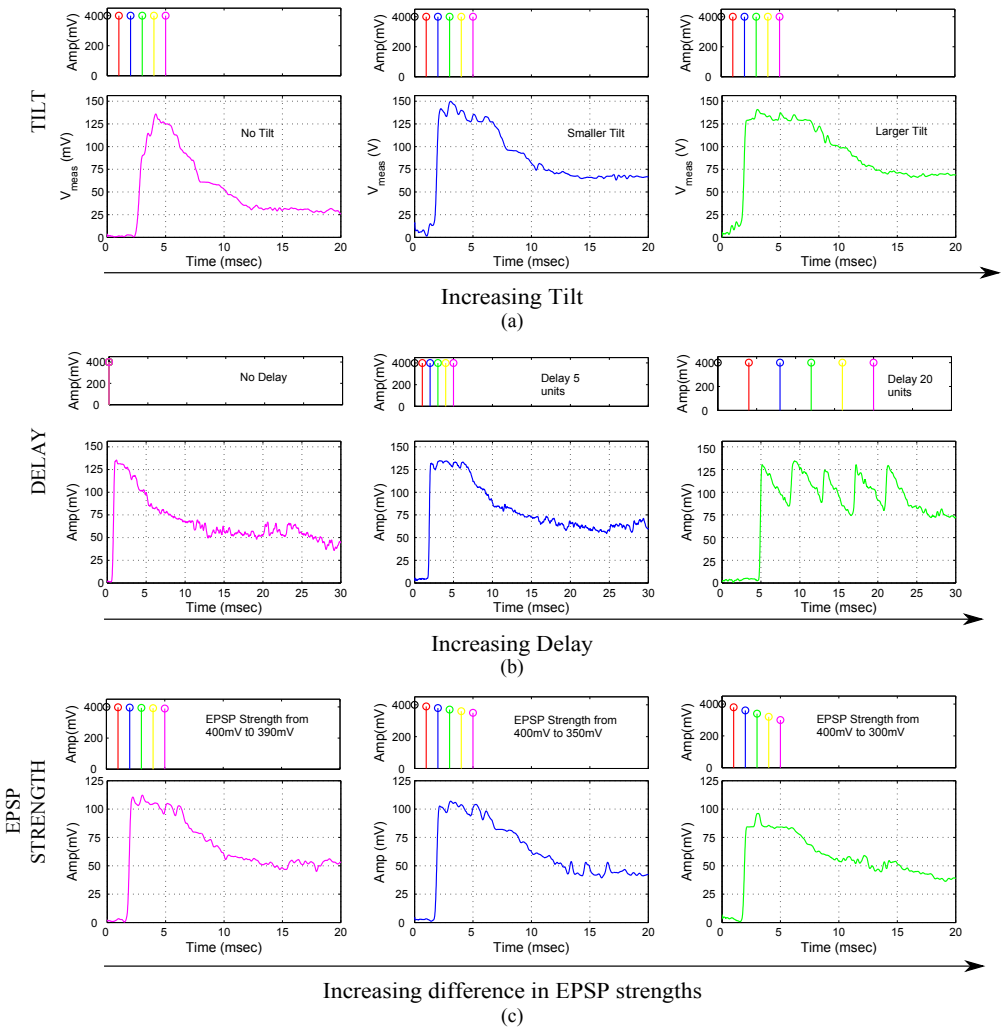


Figure 18. Experimental results for a single branch 6-tap dendrite for different metrics. All results are from the last tap of the dendrite. (a) Metric changed is the tilt of the dendrite. For subsequent figures, the tilt is increased from no tilt to a larger tilt. The diameter of the dendrite increases down the line which is achieved by increasing the conductances of the axial transistors from left to right. (b) Metric changed is the delay between EPSP inputs into each of the taps of the dendrite. In the first case we have zero time unit delay, 10 time units delay ($2ms$) for second and 20 time units delay ($4ms$)for the third diagram in the sequence. One time unit=$0.2ms$. (c) Metric changed is the difference between the EPSP strengths of the input signals. In the first case, the difference is 10mV, 50mV for the second and 100mV for the third case. We can see decreasing amplitude as the difference in EPSP strengths increases.

Now that we know that dendrites have computational significance, it is interesting to see what computational models can one build using dendrites or say a network of dendrites. One such interesting application we have discussed is classification in speech recognition.

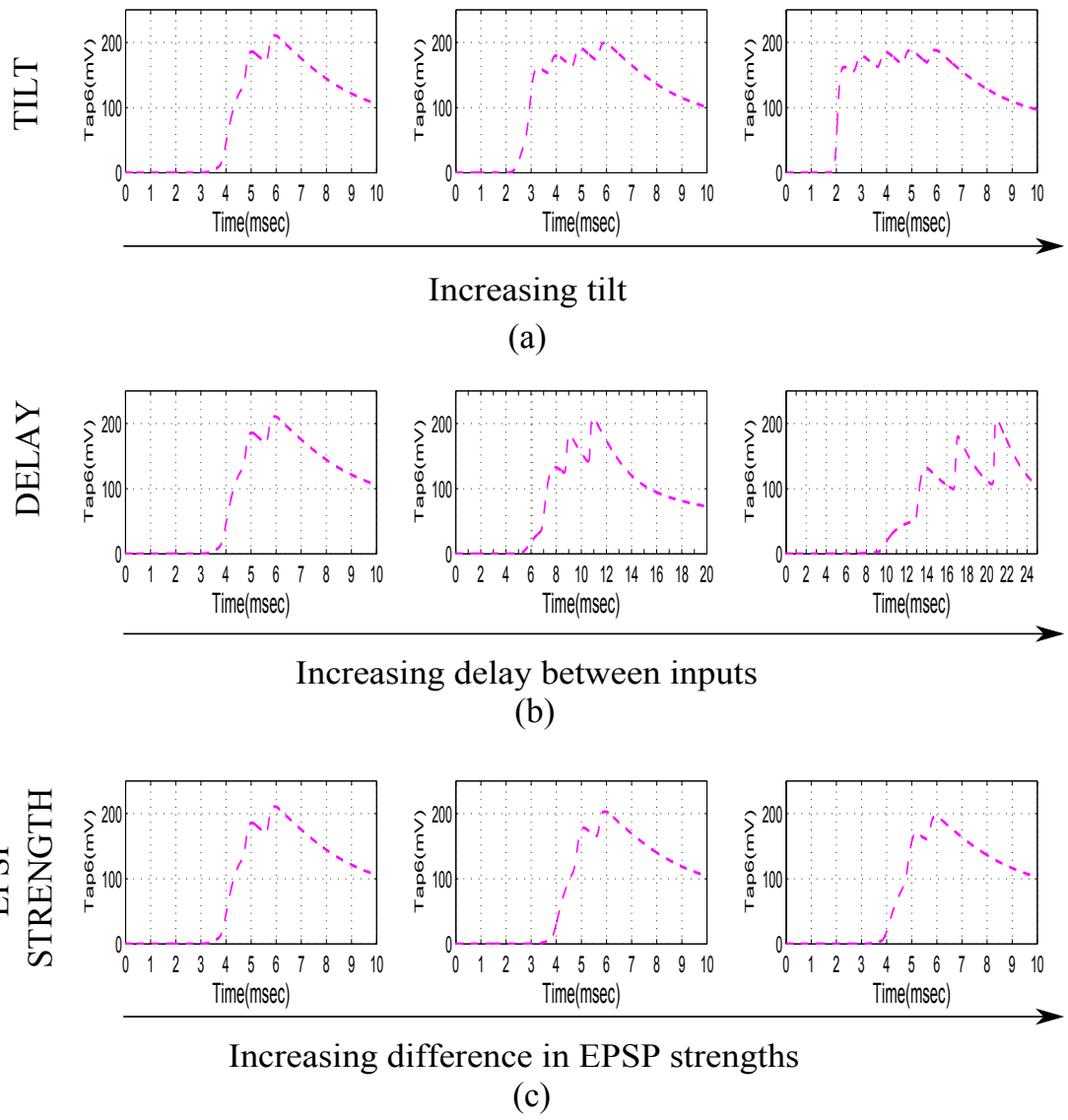


Figure 19. Simulation results for a single branch 6-tap dendrite for different metrics. All results are from the last tap of the dendrite. (a) Metric changed is the tilt of the dendrite. For subsequent figures, the tilt is increased from no tilt to a larger tilt. The diameter of the dendrite increases down the line which is achieved by increasing the conductances of the axial transistors from left to right. (b) Metric changed is the delay between EPSP inputs into each of the taps of the dendrite. In the first case we have zero time unit delay, 10 time units delay $2ms$ for second and 20 time units delay $4ms$for the third diagram in the sequence. One time unit=$0.2ms$. (c) Metric changed is the difference between the EPSP strengths of the input signals. In the first case, the difference is 10mV, 50mV for the second and 100mV for the third case. We can see decreasing amplitude as the difference in EPSP strengths increases.

We have already discussed the similarities between an HMM classifier branch and a dendritic branch[23, 17]. To test this hypotheses we implemented single dendritic branch with spatially temporal EPSP inputs to model an HMM classifier branch using a reconfigurable analog platform. We also developed a mathematical model based on the device physics of

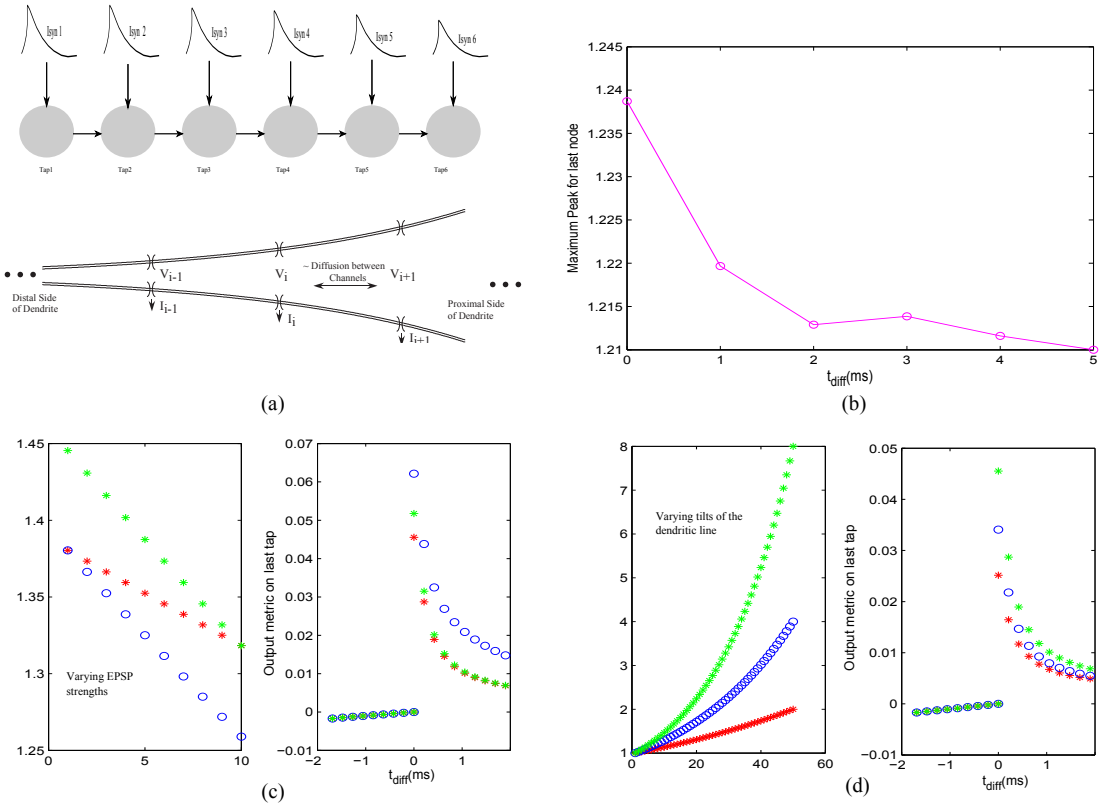


Figure 20. Experimental results, simulation results and trends observed for a single line dendrite. (a) Diagram depicting the decreasing EPSP inputs into a single CMOS dendrite line. (b)Experimental data showing change in peak to peak amplitude for a dendrite as the EPSP inputs into each of the nodes decrease down the line. (c)Change in amplitude of the output with respect to increasing difference in the EPSP amplitudes as we progress from left to right down the line(d) Change in amplitude of the output with respect to increasing difference in the tilt of the dendrite. In this experiment, the diameter of the dendrite was increased as we progress from left to right down the line.

the CMOS transistors to emulate a dendrite. In the sections ahead we will compare results seen from both models and discuss the possibilities using such models.

4.3 Hidden Markov Models

A Hidden Markov Model (HMM) is basically a state machine in which the states themselves are not observable, hence the name 'hidden'. HMMs are popular choices for classification of speech signals for speech recognition [19]. Real-world processes generally produce observable outputs, i.e. signals. Characterizing such signals into signal models can help us build practical systems like prediction and recognition systems effectively. Signal models can be classified as deterministic and statistical models. While deterministic

models characterize a specific property of a signal, statistical models characterize a statistical property of the signal [20]. For this study, I will talk about the Hidden Markov Model, which is a stochastic signal model. In a typical speech recognition model, the states would be phonemes/words and the output would be the audio signal produced by the subject. The feature of this audio signal tends to vary for different subjects. The goal of the model is to be able to correctly classify a sequence of phonemes with some tolerance.

The update rule for an HMM is given was specified in (15). Assuming $b_i[n] = 1$, $\phi_{n-1} = 0$ and assuming $a_i = 1$, (15) reduces to ,

$$\phi_i[n] = \phi_i[n-1] \tag{19}$$

Now if $b_i[n] < 1$

$$\phi_i[n] = b_i[n]\phi_i[n-1] \tag{20}$$

Thus the current state is dependent on the input probability $b_i[n]$. This shows a definite decay in the signal as we expect to see. We have previously shown how the mathematical model for HMM classification can be implemented using continuous-time analog hardware. HMMs that are considered to be discrete-time can be reformulated as wave propagating PDE that is continuous in time and space. This study demonstrates how using dendrites we can build an HMM classifier. We use Floating Gate (FG) transistors since they enable a denser programming structure and the kind of spatial-temporal dynamics that we need. We have demonstrated how dendrites with synaptic inputs perform computations similar to a Hidden Markov Model classifier branch for speech recognition as shown in Fig. 14 and Fig. 15.

4.4 Single Line CMOS dendrite

Now lets discuss a single dendrite line in more detail. Fig.16 shows a complete overview of how the CMOS dendrites are modeled and experimental results seen. Let us see in more detail, why we use EPSP signals as inputs for these dendrites. As seen in (20) where the

likelihood of a signal at the i^{th} point is a decayed version of the previous signal at the point. We have taken inspiration from Lazzaro's Analog wordspotter for classification. However, we use a different normalization technique to eliminate the decay. This interestingly, also makes such a system similar to a biological dendrite with synaptic inputs. For a continuous piece of dendritic cable we have [4, 16],

$$\tau\frac{dV(x,t)}{dt} + V = \lambda^2(x)\frac{d^2V(x,t)}{dx^2} + R(x)I_{input} \tag{21}$$

where, τ and λ are the time constant and space constant respectively. Considering that $exp(t/\tau)$ is the normalizing factor we have,

$$V(x,t) = V_1(x,t)e^{t/\tau} \tag{22}$$

where,

$$\frac{dV_1(x,t)}{dx} = \lambda^2(x)\frac{d^2V_1(x,t)}{dx^2} + R(x)I_{input}e^{-t/\tau} \tag{23}$$

$V_1(x,t)$ is the system output before normalization. Also synaptic current is given by:

$$I_{syn} \propto te^{-t/t_{peak}} \tag{24}$$

From (23) and (24) we can now see the input is similar to a synaptic current. Thus the inputs for the classifier using dendrites will be synaptic currents. The inputs can be excitatory and inhibitory in nature. In this study we assume that we have excitatory synapses as a majority of contacts on a pyramidal cell are excitatory in nature. As discussed before a dendrite doesn't have a constant diameter. This implies that for a CMOS dendrite, the conductance of the dendrite increases towards the soma, i.e. from left to right[5]. The input will thus reduce from left-right in amplitude. This indicates a decreasing strength of EPSP inputs down the line. This has been observed previously in biological dendrites [15].

This essentially signifies that the input current for the system will be similar to an EPSP input. The EPSP used to represent the input probability $b_i[n]$ are asymmetric in nature. This accounts for the normalization that is done if the inputs were symmetric Gaussian curves. It

essentially ensures that there is no sharp decay in the signal, similar to what a normalization technique would have done.

The derivation has two implications. First, we can use EPSP inputs to represent the input probabilities for phonemes. Second, the system inherently normalizes the outputs. This is similar to an HMM classifier used for speech systems.

We implemented the single dendritic line both as a CMOS circuit model and a simulation model. We found that the comparison of our experimental and simulation model was fairly close. This is demonstrated in Fig. 17 The experimental results of the single line dendrite are seen in Fig. 18.The three main parameters that govern the output of a dendrite are, namely the tilt of the line, the spatial-temporal characteristics of the synaptic inputs and the strength of the synaptic inputs. Fig. 18 and Fig. 19 demonstrate the experimental and simulation results when these parameters are varied for a single line. All results shown are for the last node of the dendrite, which signifies the output. This output indicates if the word has been detected or not. In other words the 'likelihood' that the path chosen is more likely than others.

Inputs to the PFET source

The input probabilities $b_i(t)$ are represented as log-compressed voltage signal at the dendrite node. To generate EPSP input currents into each of the dendritic nodes, we input a triangular wave voltage to the source of the pFET FG-FETs. By varying the magnitude of the triangular waves we were able to control the input current into each of the nodes of the dendrite. This can be seen in Fig.16(c). All input representations shown thus are for the inputs on the source of the transistor, that acts as synapse at every node of the dendrite. The current of a transistor is exponentially proportional to its source voltage. This enables us to generate EPSP-like inputs for the CMOS dendrite.The synaptic current is given by [4],

$$I_{syn} = I_0 e^{\kappa V_{dd}/U_T}(e^{-(V_{dd}-V_s)/U_T} - e^{-(V_{dd}-V_d)/U_T}) \tag{25}$$

where, V_s and V_d are the source and drain voltages respectively of the input transistor.

Single line dendrite results

We first implemented a single-line 6-stage dendrite. We present experimental as well as simulation results for the same. Now there are three parameters that I varied to test the behavior of dendrites for a typical speech model, namely 'tilt', delay between inputs and the difference between EPSP strengths of the input. In terms of 'tilt', two approaches were tested. First without tilt and the second with tilt. Results are shown in Fig. 18(a) and 19(a). We observed that by using tilt we could ensure that the input current would transmit more in one direction of the dendritic cable. To achieve this we increased the axial conductance of the cable down the line, such that maximum current tends to flow to the end of the cable. At every node of the dendrites we input EPSP currents in a sequence. This is the kind of behavior that we would like to emulate just as in a speech processing model, where all the phonemes/words are in a sequence and based on the sequence we classify the word/phoneme. We then varied the delay between the input EPSP signals as seen in Fig. 18(b) and Fig. 19(b). It was observed that as the delay between the inputs increases, the amplitude of the output metric decreases. This implies that as outputs are spaced farther apart there is less coincidence detection. The third parameter varied was the strength of the EPSP inputs, with the difference in EPSP strengths of the first node and the last node increasing for subsequent plots as seen in Fig. 18(c) and Fig. 19(c). The EPSP strengths near the distal end are larger than the EPSP strengths near the proximal end. Evidence for the same has been shown in biology [15]. It was observed that as the difference in amplitude was increased the amplitude of the output metric reduced. The study of these parameters showed the robustness that such a system would demonstrate in terms of speech signals. The difference in delay models the different time delays between voice signals when a word is spoken by different people. The difference in EPSP strengths ensures that the impact of all the phonemes on the output is similar for detection of a word and not dominated by just the last stage.

In Fig. 20 we studied the trends that one would observe collectively for different parameters. We varied the input sequence with respect to the time difference between signals. The output metric in this case was the difference between the output of the dendrite when all signals were present and output of the dendrite when only the last output was present. We observed that for the inputs in sequence, with varying EPSP strengths and with increasing time delay, the output metric decreased. Also, when the input sequence was reversed, with varying EPSP strengths and time delay increased the output metric was close to zero and decreased to negative values.

4.5 Analog Classifier for Word-spotting

We will now discuss the complete classifier structure. We have previously shown how the dendritic branches act as HMM branches. We have built an HMM classifier using these branches, a Winner-Take-All (WTA) circuit and supporting circuitry. We have built a YES/NO classifier structure based on the dendritic structure and neuron interaction by a group of cortical (pyramidal) neuron cells. We will simplify the modeling of a group of neuron somas and the inhibitory inter-neurons as a basic WTA block, with one winning element. You can consider the one winning element, when it transitions to a winning response as an equivalent of an output event (or action potential).

To build this network, we made a model of a dendrite, initially a single piece of cable with branch points. For the classification elements, we focus on the ability for dendritic trees to be able to compute useful metrics of confidence of a particular symbol (or concept) occurring at a particular time. This confidence metric will not only be a metric of the strength of the inputs, but also will capture the coincidence of the timing of the inputs. We would expect to get a higher metric if the $1st$, $2nd$ and $3rd$, inputs arrived in sequence, where we would expect a lower metric for the $3rd$, $2nd$ and $1st$ inputs arrived in sequence. This type of metric building is typical of HMM type networks. Simple example being if the word ‘Y’ ‘E’ ‘S’ were detected in a sequence as opposed to ‘S’ ‘E’ ‘Y’. This is

demonstrated by the simulation results as shown in Fig. 20. The metric we used here was the difference in amplitude of the last node when all inputs were present and the amplitude of the last node when only the last input was present. We observed that as we increased the timing difference between various inputs, the final metric of the line decreased as seen in Fig. 20(b). We compiled simulations to observe the effects of timing on a wider range of timing differences as shown in Fig. 20(c). We observed that the output metric decreased as we increased the timing between the inputs for a line. For the cases where we reversed the sequence, the amplitude was very close to zero. We also studied the effect of tilt on the output metric.

The network we built has two desired winning symbols either a 'YES' or 'NO'. Each symbol is represented by one or more states that indicate if a valid metric has been classified. Only the winning states would be seen as useful outputs. The useful outputs have a feedback to the input dendrite line and in effect reset the metric computations. This is implemented using a VMM structure at the end of the line and synapse at the start of the line. This is shown in Fig. 21. Each of the dendritic lines for the desired winning symbols have 5 states (dendritic compartments), where the inputs to the dendritic line represent typical excitatory synaptic inputs.

In speech/pattern recognition, signal statistics/features are the inputs to the HMM decoder. It generates the probability of the occurrence of any of the speech symbols. These signals when grouped, generate a larger set of symbols like phonemes or words [21]. Based on Lazzaro's wordspotter inputs [21], as seen in Fig. 14, we have modeled the input signals as EPSPs.

In Fig. 22 we demonstrate the detection of the word 'YES' in detail. In Fig. 23(a) and Fig. 23(b). we demonstrate detection of the word NO and then the words YES and NO in succession. Our results have demonstrated that, such a system looks similar to an HMM state machine for a word/pattern. We have based our comparisons to the analog HMM wordspotter built by Lazzaro et all [21]. We can postulate from these discussions that there

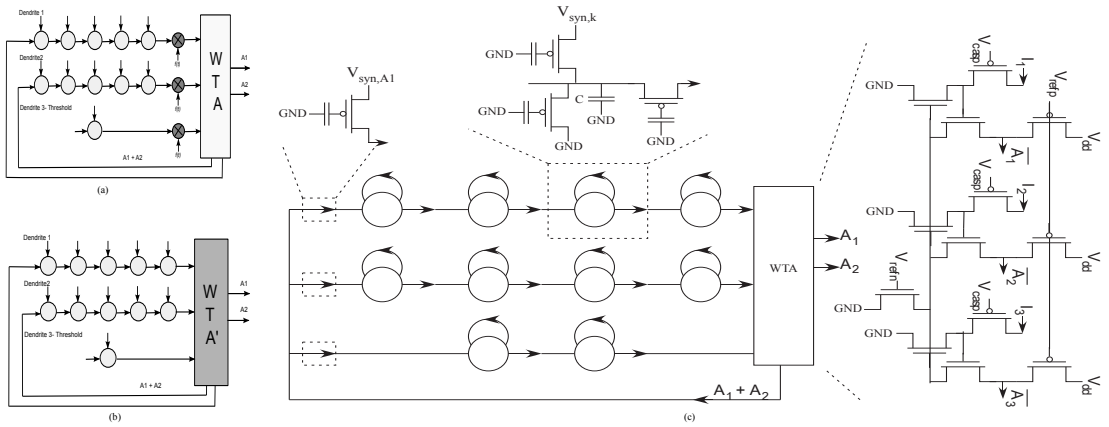


Figure 21. (a) The classifier structure with the normalization factor multiplied, $f(x) = exp{-t/\tau}$. (b) The classifier structure after normalization. (c) Detailed structure of the HMM classifier using reconfigurable floating-gate devices. There are three main structures here: The dendrite branches, the winner-take-all circuit and the supporting circuitry. The dendrite branch consists of a 5-stage dendrite for the both the branches representing YES and NO; and a single stage dendrite 3 to set the threshold current. Each dendrite has synaptic inputs at each node, which represent the phonemes of the word detected. When the line output exceeds the threshold limit, i.e. if a YES/NO is detected that the threshold loses. The supporting consists of a VMM structure and a floating-gate pFET that acts as a synaptic input at the start of the line. It also acts as reset function once a word is detected. Portion of image reprinted from [6].

are some similarities in computation done by HMM networks and a network of dendrites.

4.6 Experimental Setup

In the sections below, I will give a brief overview of the experimental setup used for the study. I used the FPAA,RASP 2.8*a* for all experimental data and the software tool MATLAB Simulink and *sim*2*spice* script to build the dendrite block.

4.6.1 FPAA review

The Field-Programmable Analog Array (FPAA) is a mixed-signal CMOS chip which allows analog components to be connected together in an arbitrary fashion. Reconfigurable Analog Signal Processor(RASP) was one of the first large scale FPAAs. It allowed us to build multiple complex circuits. The specific chip used from the family of RASP chips for this research work is RASP 2.8a [7]. It is a powerful and reconfigurable analog computing platform. It consists of thirty-two Computational Analog Blocks (CABs) and each CAB

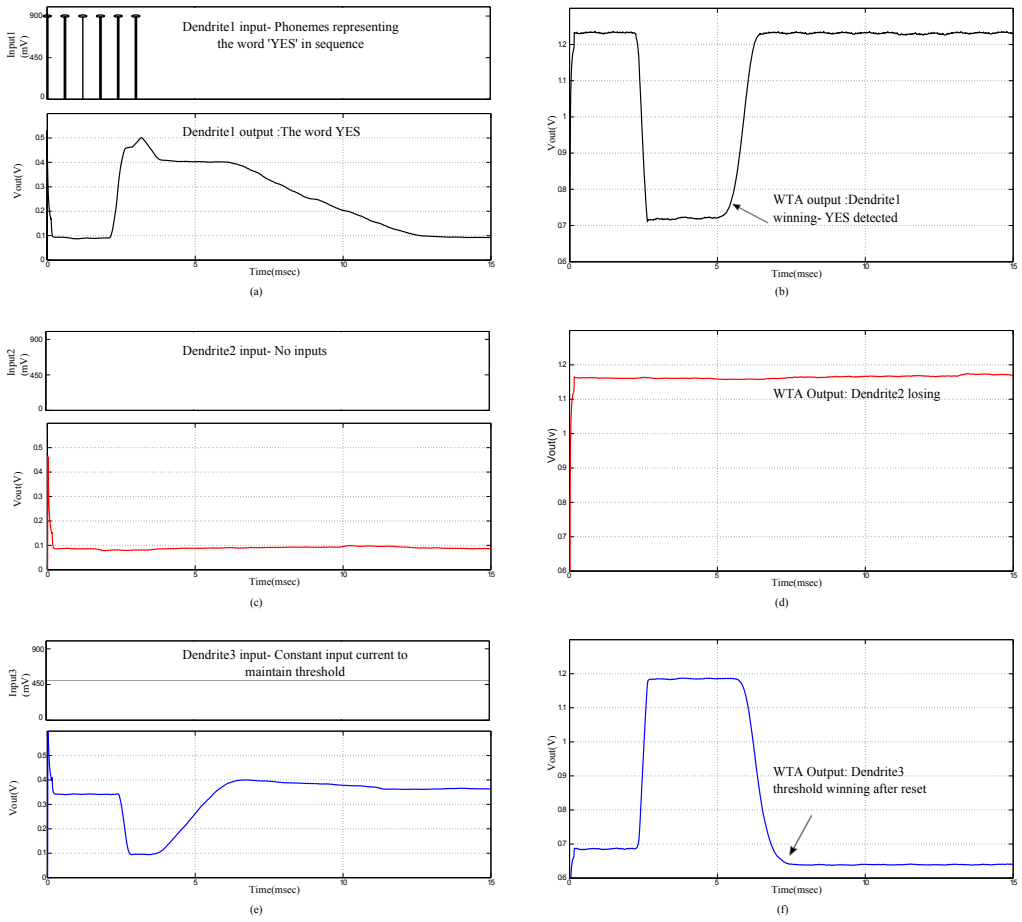


Figure 22. Experimental results for the YES/NO classifier system. The results shown are for the case when a YES is detected by the system. (a) Inputs into the nodes of the dendrites and the line output for the first dendrite. (b) Corresponding WTA output to it. A low value signifies that it is winning. (c) The line input and output for the second dendrite. (d) Corresponding WTA output. (e) The line output for the third dendrite. (f) Its corresponding WTA output. The third dendrite acts as a threshold parameter. The amplitude of the word detected on a particular line needs to be higher than the threshold to win.

consists of groups of analog circuits which include nFETs, pFETs, Operational Transconductance Amplifiers, capacitors, Gilbert multipliers, among others. These act as the computational elements which together can form complex sub-circuits that can be used to build analog computational systems. The interconnection of the CAB components is achieved by the switch matrix. It essentially consists of Floating Gate (FG) pFETs. These $50,000$ programmable elements can be used not only as programmable interconnects for routing but also as adaptive computational elements. The switch matrix allows for both local routing between CAB elements as well as global routing. Lastly, it has the programmer block,

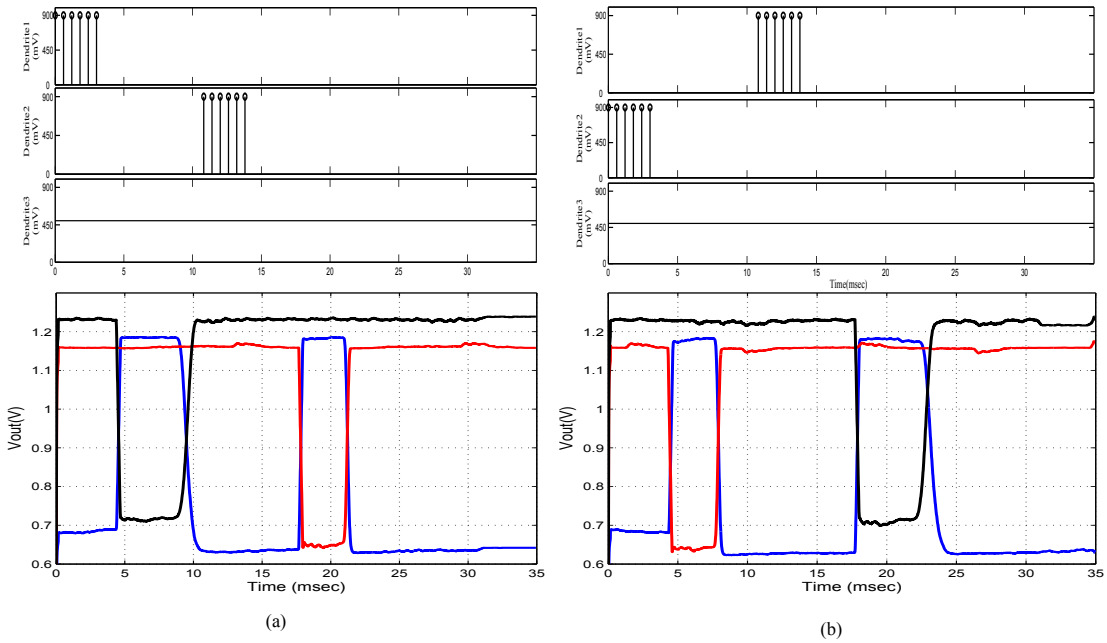


Figure 23. Experimental results for the classifier system with sequence of words detected. (a) First dendrite wins when the word YES is detected and then no is detected. The WTA input and output are shown. (b) Second dendrite wins when the word NO is detected and first dendrite wins when YES in detected.

which selectively accesses a floating-gate device on the chip and through tunneling and injection tune it to on, off or operational in between. This is not only an efficient routing scheme but also enables implementation of dense systems.

4.6.2 Dendrite on the routing fabric

Considering the fact that we required a dense network of dendrites, we used Floating Gate pFET switches. A floating-gate pFET's gate has no DC path to ground. The voltage to the gate is applied using a capacitive divider. This means that once charge is stored on the gate, it remains there without any need to apply potential directly. This also essentially means one less I/O pin. We can manipulate the charge on the gate using the processes like tunneling and injection. To place charge on the gate Fowler-Nordheim tunneling is used and to remove charge hot electron injection is used.

The key principle behind the floating-gate technology was to build systems that can

adapt and learn [8].The most exciting aspect of implementing dendritic circuits using floating-gates is that it can done in a very compact manner. As stated above, the switch matrix of the RASP 2.8a FPAA is completely made up of about 50,000 floating-gate elements. Thus huge arrays of dendrites can be made using the switch matrix. Its inherent function is to interconnect components which are similar to dendrites that are used to transmit signals from one structure to another. Modeling dendritic circuits using Floating Gates however, has few complications. The reason being the capacitive coupling from source and drain to the FG is more pronounced than regular pFETs [4].Thus, characterizing these coupling ratios is important if precision is desired. Another nonideality that arises due to indirect programming is the mismatch between the transistors that is 'programmed' versus the transistor that is actually used in the circuit. However, recently methods have been developed to characterize this mismatch [9].

Nevertheless, floating-gates enable building very compact circuits. This enables building of larger systems like HMM classifiers using CMOS dendrites. The advantage being that not only could we individually program the FG-FETs for varying levels of charge to obtain tilt easily but also could build a denser network. Also one must also take into account that neural systems are known to be inherently imprecise. Dendritic structures are not always similar and synapses are very unreliable. So one can say that this floating-gate mismatch is similar to dendrite-to-dendrite variability [4].

4.6.3 Simulink Model for simulating CMOS dendrites and FPAA configuration

Engineers have conventionally relied on digital systems like DSPs and FPGAs to implement algorithms for signal processing. A lot of software tools are available that enable and simplify this process. Thus existence of such intuitive software tools enables engineers to leverage the higher computational efficiency offered by hardware systems. Our lab has developed *sim*2*spice*, which is a tool that automatically converts analog signal processing systems from Simulink designs to a SPICE netlist [10]. It is the top-level tool in a complete chain of automation tools as shown in Fig. 6. Multiple Simulink blocks have been

implemented for different analog elements and circuits. The user has the choice of building his own analog system using basic analog elements or use existing analog blocks to build a larger system. The basic analog elements consist of the CAB elements on the FPAA .All the parameters of the block are configurable. Simulink also gives users the capability to encapsulate the final system created into a block; this ensures an ever-growing library of analog blocks that can be used. The Simulink block mainly serves two purposes: First, it converts the block-level Simulink model into a SPICE netlist which can be implemented on the FPAA. Secondly, it can also be used to run a behavioral simulation of the circuit.

Dendrite Simulink Block

The Simulink block simulates the behavioral characteristics of the dendrite structure given input/s. This provides the user an insight to the working of the dendritic circuit when implemented using the FPAA. The MOSFET parameters used are based on the MOSFETS present on the FPAA. It is characterized by coupled ordinary differential equations (ODE) and solved using the ODE solver ode-45. The model has been tested for both static as well as time-varying inputs and has given reasonable results. For this study we have used EPSP signals as inputs for the block. Consider a dendritic line as given in Fig. 15(c), with n number of nodes. If we assume that the axial and leakage conductances are the same throughout the line, the voltage at each node can be calculated using the following coupled ODE [4],

$$\begin{aligned}\frac{d\vec{V}}{dt} = \frac{1}{C}(a_1 \cdot I_{inj} &+ k_1'(e^{a_2\cdot\vec{V}/U_T} - e^{a_3\cdot\vec{V}/U_T}) \\ &+ k_1'(e^{a_4\cdot\vec{V}/U_T} - e^{a_5\cdot\vec{V}/U_T}) \\ &+ k_2(e^{a_6\cdot\vec{V}/U_T} - e^{E_k/U_T}) + I_{bias})\end{aligned} \tag{26}$$

For tilt, I varied the parameter k_1' as it is proportional to axial conductances.

4.7 Classifier: Computational efficiency

A major advantage that analog systems have over digital systems is computational efficiency. This can be seen in Table 1. The unit used to compare computational efficiency

Table 1. Comparing computational efficiency of Digital, Analog and Biological systems

Computing Type	Efficiency	Energy/MAC
Digital (DSP)	< 10MMAC/mW	> 100pJ
Analog SP (VMM)	10MMAC/μW	100fJ
Neural Process	> 10MMAC/pW	< 0.1aJ

is Multiply ACcumulates (MAC) per second. The energy efficiency at a given node of the system depends on the bias currents, supply voltage and also the node capacitance.

The time constant τ is the product of the node capacitance and conductance. Energy E is the product of power and time. Now the bias current I_{bias} for a dendrite node is given by,

$$I_{bias} = (V_{rest} - E_k)G; \tag{27}$$

where, V_{rest} is the resting potential, E_k signifies the voltage of a potassium channel and G is the axial conductance. We can write this as,

$$I_{bias} = (V_{rest} - E_k)\frac{C}{\tau} \tag{28}$$

Also, power is the product of voltage across the node and current into the node.

$$Power = V_{dd}(V_{rest} - E_k)\frac{C}{\tau} \tag{29}$$

V_{dd} is the supply voltage, V_{rest} and E_k the internal and external potentials of the leak channel.

$$Energy = V_{dd}(V_{rest} - E_k)C; \tag{30}$$

Now for a single node of an HMM classifier, we have 2 MAC/sample. Assuming $\tau \sim delay$, which at a given node is approximately $1ms$. Thus,

$$Energy/MAC = \frac{1}{2}V_{dd}(V_{rest} - E_k)C \tag{31}$$

From the equation it is evident that major factors contributing to energy efficiency in this case is the node capacitance. As we scale down the process used, this value will reduce. Currently the node capacitance on the chip we used was $1pF$. If we further scale down the

Table 2. Comparing computational efficiency depending on load capacitance

Process	Capacitance	V_{dd}	Energy/MAC
Analog	$1pF$	2.4V	$12.00fJ/MAC$
Analog	$10fF$	2.4V	$0.12fJ/MAC$
Biological	$1fF$	$200mV$	$0.02fJ/MAC$

process used, this number will also reduce. This effectively means higher computational efficiency. A decrease to $10fF$ itself will give us an improvement of 2 orders of magnitude as seen in Table 2.

4.8 Broader Impact

The broader impact of such a system is two-fold. First, this system is an example of a computational model using bio-inspired circuits. Secondly, the system proposes a computationally efficient solution for speech-recognition systems using analog VLSI systems. As we scale down the process, we can get more efficient and denser systems. We can also address how synaptic learning can be implemented and classification systems be trained. It is known that even the world's fastest supercomputers cannot match the computational efficiency that biological systems have. It is evident from the computational efficiency discussions that, analog systems are clearly a better choice for higher computational efficiency and lower costs. This calls for greater effort to build such systems. Reconfigurable/programmable analog systems open a wide range of possibilities in demonstrating biological processing and also for signal processing problems. This will not only enhance our understanding of biological processes but will also help us design more efficient systems that have tremendous computational abilities.

4.9 Conclusion

We have discussed the similarities between the field of neuro-biology, silicon devices and HMMs. We have thus postulated similarities between Hidden Markov Models and Neural Systems. This work also demonstrates a computational model using bio-inspired CMOS

dendrites. We built an HMM classifier that was used for wordspotting. We have demonstrated a YES/NO decision structure using bio-physically inspired CMOS dendrites. We have also found that this implementation is computationally efficient and a more viable solution for speech recognition devices as opposed to its digital counterpart. We have demonstrated a single dendritic line with 6 compartments, with each compartment having a single synaptic input current. We have seen the behavior of a single dendrite line by varying three parameters namely the tilt, the temporal relations between inputs and the strength of the EPSP inputs. The effects of tilt which enabled coincidence detection were studied. We have also seen the functioning of the WTA block with dendritic inputs and how the feedback helps initiate the reset after a word/phoneme is detected. We also build a Simulink dendritic model and simulated the output for time-varying inputs. This demonstrated how such a network would behave if inputs were in a sequence or if they were reversed. We have demonstrated a computational model using dendrites and also demonstrated a block that can be used to build computationally efficient speech recognition systems. This technology is attractive especially for implantable devices.

CHAPTER 5

CONCLUSIONS AND FUTURE DIRECTIONS

In this study, I have demonstrated that it is possible to create a voltage-mode CMOS dendrite which maintains certain properties of linear cables. With this as a fundamental building block in Neuromorphic circuits, we can explore more interesting topologies. Dendrites with active channels can be implemented that act as classifiers, support back-propagation, perform coincidence detection and assist many other neural circuits. This will enable us to develop complex neural systems.

I developed a dendrite Simulink model that can be used to simulate the behavior of a single dendritic line. The results from the simulations were found to be qualitatively similar to the results obtained experimentally. The model was further used to implement a dendritic branch similar to an HMM branch for the classifier system. The Simulink block was also used to generate A SPICE netlist using the *sim*2*spice* tool [10] to implement the circuit on the FPAA.

5.1 Summary of thesis results

In the first chapter I provided an overview of neural systems and how they can be emulated using silicon. We also had discussions about dendrites as computational subunits and not merely conductors. We discussed the various developments in technology that has made the implementation of biological circuits possible. We discussed how understanding the interaction among neural elements is a bigger problem than just the function of the elements individually.

The second chapter focused primarily on the tools that our lab has developed that has enabled us to build reconfigurable analog systems. We had a brief overview of field programmable analog arrays and floating gates. We then discussed the process flow that is adopted to compile Simulink blocks down to the FPAA. The tools discussed are *sim*2*spice*,

GRASPER and RAT. These tools were pivotal for my experiments.

The third chapter described the modeling and implementation of voltage-mode CMOS dendrites using reconfigurable systems. I demonstrated that it is possible to create a voltage-mode CMOS dendrite which maintains certain properties of linear cables. In this chapter, I primarily developed the Dendrite Simulink macro-model. The model can be used for both behavioral simulations as well as generating a SPICE netlist to implement the dendrites on the FPAA. I also performed steady-state and dynamic experiments and concluded that for very small signals, the CMOS dendrite was qualitatively similar to linear cables.

The fourth chapter detailed the use of dendrites as a computational block used for classification. I talked about the co-relations between Neural Systems, CMOS transistors and HMM networks. I compared the similarities between an HMM branch and a dendrite branch. I also presented experimental and simulation data to support this hypothesis. I also demonstrated results for an HMM classifier system using CMOS dendrites to build a YES/NO wordspotter.

The fifth chapter summarized the work done and discusses the future possibilities and applications.

5.2 Future Directions

This study covered the modeling and implementation of CMOS dendrites using a reconfigurable analog platform. We have also demonstrated how we can build computational models using dendrites.

5.2.1 Macromodeling

Macromodeling is basically the functional design of a block. It essentially simulates the behavior of a block without overly going into details or second-order effects. With the large set of analog blocks that we have developed, it is imperative that we implement the behavioral models of these blocks. This has namely two advantages: First, it gives the user a fast and simple way to see the functioning of a block. Secondly, it gives the user flexibility

to see if the designed system works when designing larger systems. This calls for Simulink blocks that can demonstrate accurately if a system works or not. In my current study itself, I was able to see the advantages of having a macromodel for the dendrite. It enabled me to predict the trends given a certain set of parameters for the dendrite model. I thus plan to develop the macromodels of such Neuromorphic elements like the dendrite and other analog blocks. With the basic functionality in place, we can further add complexity to the models to make them more robust [10].

5.2.2 Larger computational structures

In chapter 4 I discussed how the node capacitance can influence the computational efficiency of the system. Though this number is significantly less as compared to digital systems, there is scope to minimize this further as we scale down the process. Currently we have fabricated a chip in the $130\mu m$ process with node capacitances roughly $10fF$. Considering the current implementation using the RASP 2.8a has node capacitances of about 1pF, we will see an efficiency increase by two orders of magnitude. Also the architecture of the chip allows denser circuits and thus we can build larger computational models using it. I plan to implement larger classifier structures using this chip. Also if we are classifying words, what might those inputs be (phonemes) and how we can generate them using filter banks and WTA based classifier for basic auditory signals. Also, once can begin to address questions on how to I also envision that we can begin addressing the issues of learning, building dendrites with active channels to introduce non-linearity, branched dendrites and many other interesting topologies.

REFERENCES

[1] W. F. Allman, *Apprentices of Wonder: Inside the Neural Network Revolution*. Bantam Books, 1989.

[2] C. Mead, *Analog VLSI and Neural Systems*. Reading, MA: Addison-Wesley, 1989.

[3] A. Polsky, B. W. Mel, and J. Schiller, "Computational subunits in thin dendrites of pyramidal cells," *Nature Neuroscience*, vol. 7, pp. 621–627, 2004.

[4] S. Nease, S. George, P. Hasler, and S. Koziol, "Modeling and Implementation of Voltage-Mode CMOS Dendrites on a Reconfigurable Analog Platform," *IEEE Transactions on Biomedical Circuits and Systems*, accepted for publication.

[5] E. Farquhar, D. Abramson, and P. Hasler, "A reconfigurable bidirectional active 2 dimensional dendrite model," *Proc. ISCAS*, vol. 1, pp. 313–316, 2004.

[6] E. Farquhar, D. Abramson, and P.Hasler, "A Reconfigurable Bidirectional Active 2 Dimensional Dendrite Model," *IEEE International Symposium on Circuits and Systems*, vol. 1, pp. 313–316, 2004.

[7] A. Basu, S. Brink, C. Schlottmann, S. Ramakrishnan, C. Petre, S. Koziol, F. Baskaya, C. Twigg, and P. Hasler, "A Floating-Gate-Based Field Programmable Analog Array," *IEEE Journal of Solid-State Circuits*, vol. 45, pp. 1781–1794, 2010.

[8] P. Hasler, *Foundations of Learning in analog VLSI*. PhD thesis, California Institute of Technology, 1997.

[9] S. Shapero and P. Hasler, "Precise Programming and Mismatch Compensation for Low Power Analog Computation on an FPAA," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 1, p. 1, submitted for review.

[10] C. Schlottmann, C. Petre, and P. Hasler, "A High-Level Simulink-Based Tool for FPAA Configuration," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. Issue:99, pp. 1–1, 2010.

[11] S. Koziol, C. Schlottmann, A. Basu, S. Brink, C. Petre, B. Degnan, S.Ramakrishnan, P. Hasler, and A. Balavoine, "Hardware and software infrastructure for a family of floating-gate based FPAAs," *IEEE International Symposium on Circuits and Systems*, vol. 1, pp. 2794–2797, 2010.

[12] F. Baskaya, D. Anderson, P. Hasler, and S. K. Lim, "A generic reconfigurable array specification and programming environment," *Circuit Theory and Design, 2009. ECCTD 2009. European Conference on*, vol. 1, pp. 619–622, Aug. 2009.

[13] D. A. J.D. Gray, C.M. Twigg and P. Hasler, "Characteristics and programming of floating-gate pFET switches in an FPAA crossbar network," *IEEE International Symposium on Circuits and Systems*, vol. 1, pp. 468–471, 2005.

[14] M. London and M. Hausser, "Dendritic Computation," *Annual Review of Neuroscience*, vol. 28, pp. 503–532, July 2005.

[15] B. W. Mel, "What the Synapse Tells the Neuron," *Science*, vol. 295, pp. 1845–1846, 2002.

[16] C. Koch, *Biophysics of Computation*. New York, NY: Oxford University Press, 1999.

[17] P. Hasler, S. Koziol, E. Farquhar, and A. Basu, "Transistor Channel Dendrites implementing HMM classifiers," *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on*, vol. 1, pp. 3359 – 3362, 2007.

[18] R. P. Lippmann, E. I. Chang, and C. Jankowski, "Wordspotter training using figure-of-merit back-propagation," *Proceedings International Conference on Acoustics, Speech, and Signal Processing*, vol. 1, pp. 389–392, 1994.

[19] B. H. Juang and L. R. Rabiner, "Hidden Markov Models for Speech Recognition," *Technometrics*, vol. 33, pp. 251–272, 1991.

[20] L. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proceedings of the IEEE*, vol. 77 (2), pp. 257–28, 1989.

[21] J. W. John Lazzaro and R. Lippmann, "A Micropower Analog VLSI HMM State Decoder for Wordspotting," *Advances in Neural Information Processing Systems 9*, vol. M. C. Mozer, M. I. Jordan, and T. Petsche, Eds. Cambridge, Massachusetts: MIT Press, p. 727733, 1996.

[22] Y. Wang and S.-C. Liu, "Input evoked nonlinearities in silicon dendritic circuits," *IEEE International Symposium on Circuits and Systems*, vol. 1, pp. 2894 – 2897, 2009.

[23] P. Hasler, P. Smith, D. Anderson, and E. Farquhar, "A Neuromorphic IC Connection Between Cortical Dendritic Processing and HMM Classification," *IEEE 11th Digital Signal Processing and 2nd Signal Processing Education Workshop*, pp. 334–337, 2004.